

# תקשות טורית UART

## הקדמה:

תקשות טורית (UART) היא שיטה נפוצה ו פשוטה להעברת מידע בצורה טורית (סידורית) בין שני התקנים דיגיטליים, כגון בין מיקרו-בקר לבין מחשב או מודם. זהו אחד מפרוטוקולי התקשורת הבסיסיים ביותר הנמצאים בשימוש בתחום האלקטרוניקה הדיגיטלית.

בבסיסה, תקשורת UART מורכבת וכוללת פרוטוקולים המאפשרים שידור וקליטה של מידע בצורה אמינה ויעילה. היא משתמשת בשני חוטי תקשורת עיקריים: "קבל (RX)" ו- "שלח (TX)".

## מהי תקשורת UART?

תקשות טורית (UART) היא שיטה פשוטה ונוחה להעברת מידע בין שני התקנים דיגיטליים, למשל בין מיקרו-בקר כמו ארדואינו למחשב.

בקשות UART המידע מועבר בצורה **סיביות** (זרמים מתחלפים גבירות ונדירות - ראו תמונה בהמשך) וMOVUBROTות בחוותי תקשורת נפרדים בין שני המכנים. כל מכשיר מחובר עם שני פינים לתקשורת - **קלט (RX)** ו**פלט (TX)**.

כדי לשדר מידע, המכשיר (השולח) שולח את הביטים על גבי קו ה- TX. הצד השני, המתקבל מזין לאותו קו ומזהה את המידע בכניסה ה- RX. שלו ופענה אותו בחזרה למידע דיגיטלי.

עקרונות חשובים ב-UART הם **קצב התקשורת** - baud rate - כמה ביטים בשנייה, (**אורץ קבוע** - 8 ביטים בשנייה, **בית התחלה** (סינכרון לתחילת העברת מידע), **בית סיום** (סימון סיום ההעברה).

היתרונות הגדולים של תקשורת טורית הם פשוטות וミニימליות בחומרת המכשיר, מה שמאפשר להשתמש בה ליישומים רבים עם מגבלות תוכנה וחומרה.

לכן UART היא אחת השיטות הנפוצות להעברת נתונים בין מיקרו-בקרים לבין מחשבים במגוון יישומים כמו התקני אינטרנט של הדברים, מערכות בקרה תעשייתית, רובוטיקה ועוד.

**Łסיכום:** תקשורת טורית מאפשרת לנו לשדר ולקלוט נתונים באופן אמין ופשוט יחסית בין כל סוגי התקנים הדיגיטליים.

## מאפיינים תקשורת טורית:

- **תקשות סינכרונית** (סיראלית) - המידע מועבר כבית, בית אחרי בית, על גבי קו תקשורת יחיד.
- **א-סינכרונית** - אין שעון משותף, כל התקן פועל לפי שעונו הפנימי.
- **תקשות דו-כיוונית** - יכולה לשדר ולקלוט נתונים.
- **חד כיווני (Simplex)** - שידור רק בכיוון אחד.
- **חצי דו כיווני (Half-Duplex)** - בכל רגע נתון, רק צד אחד משדר והשני קולט ולהפך, זהו האופן הנפוץ ביותר.
- **שידור וקליטה בקצב זהה** (קבוע מראש לדוגמה 9,600 ביט לשנייה).
- **מבנה מוגדר היטב** של מסגרת הנתונים (פרויימים) - סיביות התחלה, נתונים, סיביות עצירה.
- **דרישות חומרה פשוטות** - בדרך כלל 2 קווים בלבד כיוון (TX ו-RX).
- **תמך חומרה פשוט וסטנדרטי** (למשל UART 16550 במחשבים).

- שימוש נרחב במערכות מובנות (מחשב) ובתקשורת בין מכשירים.
- **פרוטוקולי חיבור** התקני קטן כמו מודמים, מדפסות, מודמים ועוד.
- **פרוטוקולים נפוצים הבנויים מעל UART :** RS-232, RS-422, RS-485 .

#### מבנה חבילת נתונים בתקשורת טורית:

כאשר מתכוון (למשל מיקרו-בקר) מעוניין לשדר נתונים, הוא בונה חבילה נתונים מבנה מוגדר הכולל מספר שדות :

- **בית התחלה (Start Bit)** - ירידה מ-'1' ל-'0' לוגי, המסמך את תחילת השידור.
- **נתונים** - המידע עצמו במסגרת של ביטים (בדרך כלל 8-5 ביטים).
- **בית זוגיות (Parity Bit)** - אופציונלי לבדיקת שגיאות.
- **בית עצירה (Stop Bit)** 1 לוגי, המסמך את סוף השידור.

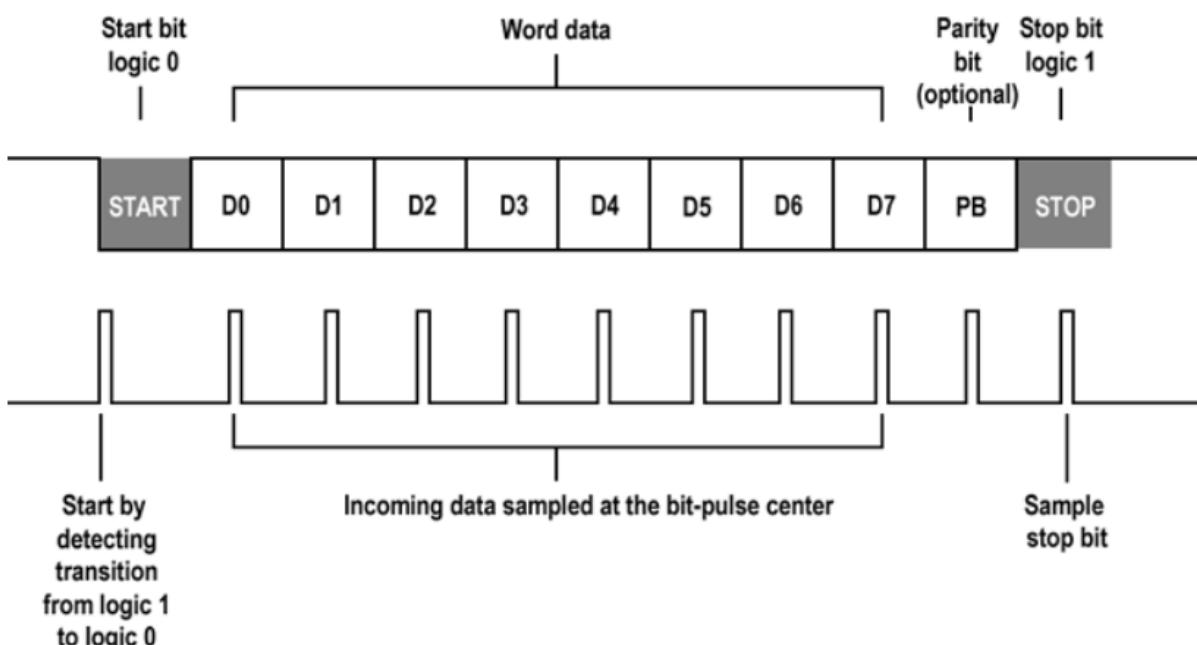
#### דוגמה:

במידה ונרצה לשЛОוח את התו F (המספר 63 בעשרוני) ייבנה מסר באורך 8 ביטים :

**Start Bit (0) → 00111111 → Stop Bit(1)**

כלומר, **בית התחלה אחד (0)**, אחריו (מימין לשמאל) **הביטים של הנתון (11111111)**, ולבסוף **בית עצירה אחד (1)**.

בין כל שידור של חבילה כזו יש מרווה זמן קצר לא שידור, כך שהצד מקבל יוכל לזהות היטב את בית ההתחלה הבא ובכך את תחילת הנתונים החדשים.



ב**בית זוגיות (optional)** .

ב**בית התחלה** .

ב**בית סיום** .

ה**מעיך** שאנו שולחים .

- תחילת שידור – מעבר מהמתה גובה 0 לוגי למתח הנמוך (0 לוגי) שמסמן את תחילת העברת המידע.**
- נתונים מגיעים בפולסים של ביטים.**
- סיום שידור.**

#### **בית זוגיות (Parity Bit):**

beit zogiyot meshem loziohi shgaiot basisi b'tekshoret torriah. hoa mosif beit nosaf l'masgerat hantoinim, kach shmasper habitiim um urk 1 yihya tamid zogi (zogiyot zogiyah) au ai-zogi (zogiyot ai-zogiyah). am b'kliyut hamidu masper habitiim um urk 1 la mataim l'sog hzogiyot shnabu, haklal kol lehnih shairura shgaiyah b'shidur. um zat, beit zogiyot la maafshar ltakan shgaiot ao lozohot shgaiot morbohot b'bitim, l'kun hoa shiota basisit belbad looyidaa shlemot hantoinim.

#### **תקשורת טוריית ב-ESP32:**

h-ESP32-bnigod laardoiaino avnu, mazoid b'masfer ychidot tekshoret torriah (UART). b'dorh call, yesh lo shelsha mmashki UART, maha shhafek otovu l'mataim b'miyyad l'projektiim zdorshim chibor l'masfer haknanim chizoniim camo modoli GPS, chiyuni tmaprotora, meschi LCD v'oud.

#### **הבדלים העיקריים בין תקשורת טוריית בארדואינו ל-ESP32:**

- **מספר ייחדות UART :** בעוד שלארדואינו אוננו יש רק ייחדת UART אחת (בטיסות 0 ו- 1), ל- ESP32 יש שלושה ממתקי UART : (UART0, UART1, UART2) זה מאפשר לפתח פרויקטיים מורכבים יותר המשמשים במספר רב של מכשירים הפועלים בשיטה טוריית.
- **חופש בבחירה הפינים :** ה- ESP32 מאפשר לתוכנת לבחור באופן חופשי את הפינים (GPIO) שיישמו לכל ייחודה UART, בזכות יכולת מולטיפקסינג הפינים. המשמעות היא שניתן להגדיר את פיני ה- TX ו- RX של כל UART לכל פין GPIO פניו בלוח, דבר המעניין גמישות רבה בתכנון החומרה.
- **פונקציונליות :** כל ייחדת UART ב- ESP32 היא עצמאית לשלוטין וכוללת מאגרים (buffers) גדולים לקליטה ושידור. זה מבטיחה שהמידע לא יאבז גם כאשר גם יש עומס גדול של נתונים, ומאפשר שימוש ב- UART0 (שמוחבר גם ל- USB) לצורך ניפוי באגים (debug) בתוכנה, תוך כדי שימוש ב- UART1 ו- UART2 לצורך תקשורת עם רכיבים אחרים.

#### **קצב שליחת נתונים:**

ano mgadrim at katzb shelicht hantoinim b'tekshoret torriah b'pkoda achot belbad b'fonkzia void Setup באמצעות הפוקודה :

**Serial.begin(9600);**

### רשימת קצבי השידור הנפוצים בתקשורת טוירית (UART):

|                     |
|---------------------|
| • 300 בית לשניה     |
| • 600 בית לשניה     |
| • 1,200 בית לשניה   |
| • 2,400 בית לשניה   |
| • 4,800 בית לשניה   |
| • 9,600 בית לשניה   |
| • 19,200 בית לשניה  |
| • 38,400 בית לשניה  |
| • 57,600 בית לשניה  |
| • 115,200 בית לשניה |
| • 230,400 בית לשניה |
| • 460,800 בית לשניה |
| • 921,600 בית לשניה |

ניתן לראות שיש מגוון רחב של קצבי שידור, החל מ-300 בית לשניה במערכות ישנות יותר, ועד למיליאוני ביטים לשניה במערכות טויריות מהירותים יותר.

**הकצבים הנפוצים ביותר הם:** 9,600 בית לשניה ו-115,200 בית לשניה.

קצב השידור קבוע מראש ומהיב להיות זהה בין שני ה התקנים המתקשרים כדי לאפשר קליטה נכונה של הנתונים.

### רכיבים שעובדים בתקשורת טוירית:

- **מודולי בלוטוס** - HC-05, HC-06
- **מודולי WiFi** - ESP8266, ESP32
- **מודולי LoRaWAN/LoRa** - לתקשורת לטוחה רחוק
- **מודולי תקשורת סלולרית** - SIM900, SIM800L
- **מודולי GPS** - NEO-6M, UBLOX
- **צגי OLED ו-LCD** טוריים (I2C)
- **חיישני טמפרטורה, לחות, אור ותנוועה**
- **מודולי קוראי RFID/NFC**
- **מודולי מצלמות טויריות**
- **כרטיסי SD וקוראי כרטיסי SD**
- **מדפסות ומפענחים טויריים**

### שליחת נתונים בתקשורת טורית:

ניתן לשולח נתונים בתקשורת טורית באמצעות הפקודה :

**Serial.print();**

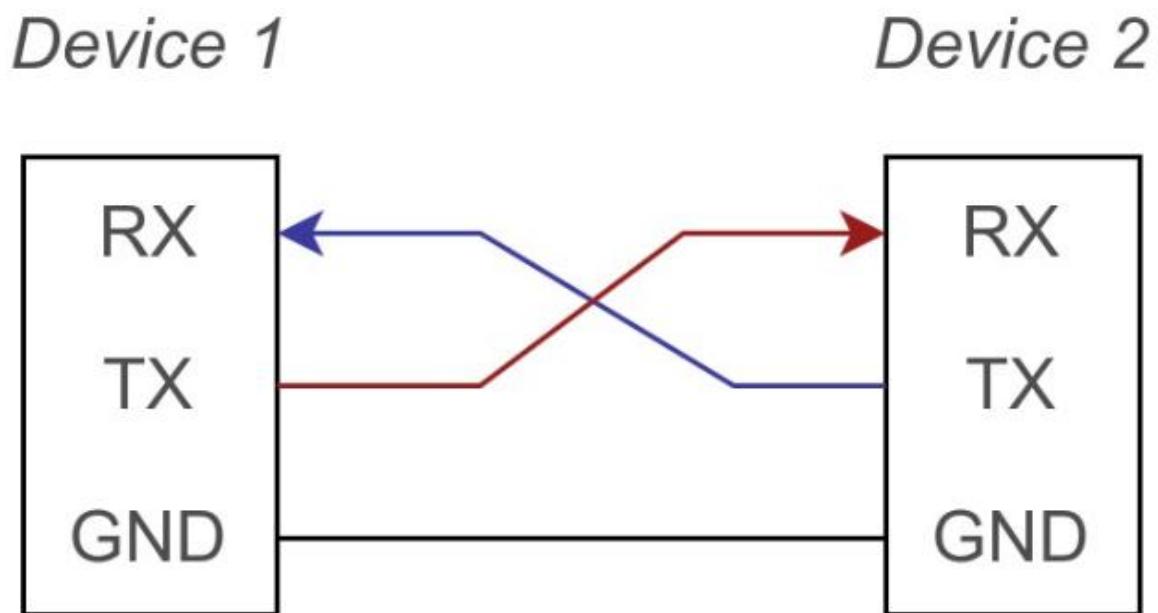
כל מה שנרשם בין הסוגרים () עם מראכות ישלח בתוUr ערך של משתנה. בפקודה זו אין ירידת שורה. במידה ונרצה לרדת שורה אנו צריכים להשתמש בפקודה :

**Serial.println();**

### חיבור בין 2 רכיבים בתקשורת טורית:

בחיבור בין שני התקני UART נדרשים לפחות 3 חוטים - TX של צד אחד מתחבר ל-RX של הצד השני, וחוט האركה משותף (GND).

לעתים נדרש גם חיבור של מתח ההזנה (Vcc) אם אחד ההתקנים מספק כוח לשני. יש לוודא התאמה של רמות המתח בין שני הצדדים (3.3 או 5V).



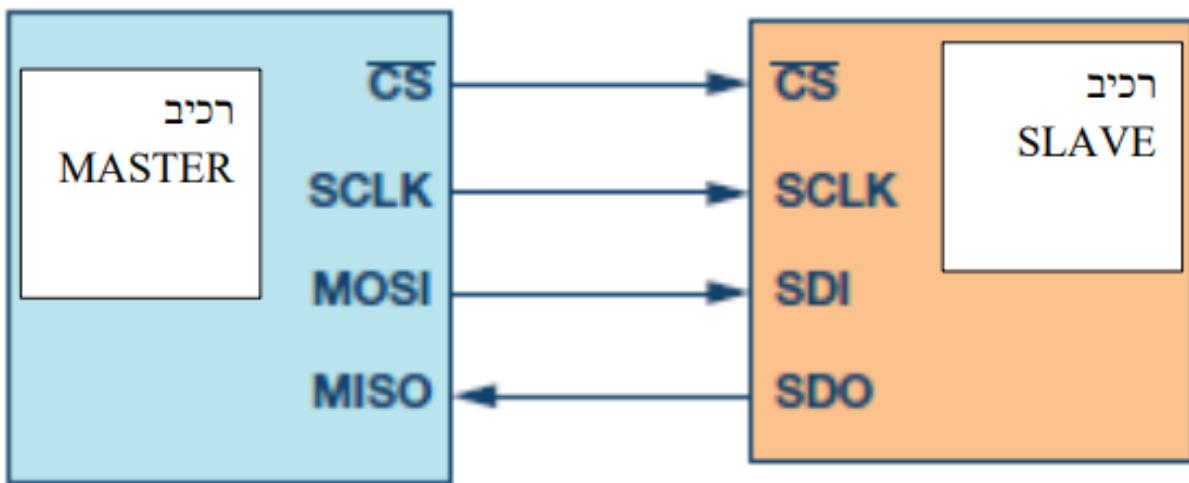
## תקשורת SPI

התקשורת נוצרה על ידי חברת מוטורולה ב 1979 עם יציאת המיקרו פרוססור של חברת מוטורולה שנקרא 68000.

השם SPI הוא – Interface Peripheral Serial – ממשק טורי היקפי . זהה תקשורת טורית סינכרונית כי יש בה שעון שמסנכרן את הכניסה/ היציאה הנתונים.

דוגמאות לרכיבי SPI הם מתגים, זיכרונות, רכיבי שמע להקלטה/השמעה , ממירים למיניהם (ADC), תצוגות לדיס, TFT ועוד.

באיור הבא מתואר חיבור של תקשורת טורית SPI בין רכיב MASTER ורכיב SLAVE.



### באיור רואים שבתקשורת SPI יש 4 קווים:

- .1 : קו הנtauו מהMASTER (המיקרו בקר ) אל העבד (ברכיב העבד השם הוא Master out slave in – MOSI ).  
.2 : קו הנtauו הטורי מהעבד אל המMASTER. ברכיב העבד השם הוא Master in slave out – MISO .  
.3 : שני האותיות MOSI ו- MISO מסונכרנים בעזרת קו השעון הטורי .  
.4 : זוהו קו נוסף שדרכו אנו בוחרים את העבד ה- slave , בעזרת קו זה המיקרו בקר מודיע לאיזה עבד הוא פונה. הקו פועל בנמוך – LOW ACTIVE . שמota נוספים לקו him – CS – בבחירה רכיב – Enable , אפסור וועוד .

### תהליך התקשורת מתבצעת בשלבים הבאים:

- כאשר ה-MASTER מתחילה תקשורת עם ה-SLAVE הוא מוריד את קו בחירת הרכיב עבד (Select slave) ל-0.
- ה-MASTER שלוח בקו ה- MOSI בית אחרי בית, כאשר כל בית מסונכרן בעזרת פולס שעון – SCLK – שמייצר ה- MASTER לתוך ה- SLAVE .
- הביטים הנשלחים נמצאים ברגיסטר הזהה הנמצא בתוך ה- SLAVE .
- הסyncronו לתוך ה- SLAVE יכול להיות בעליית פולס שעון או בירידה שלו.

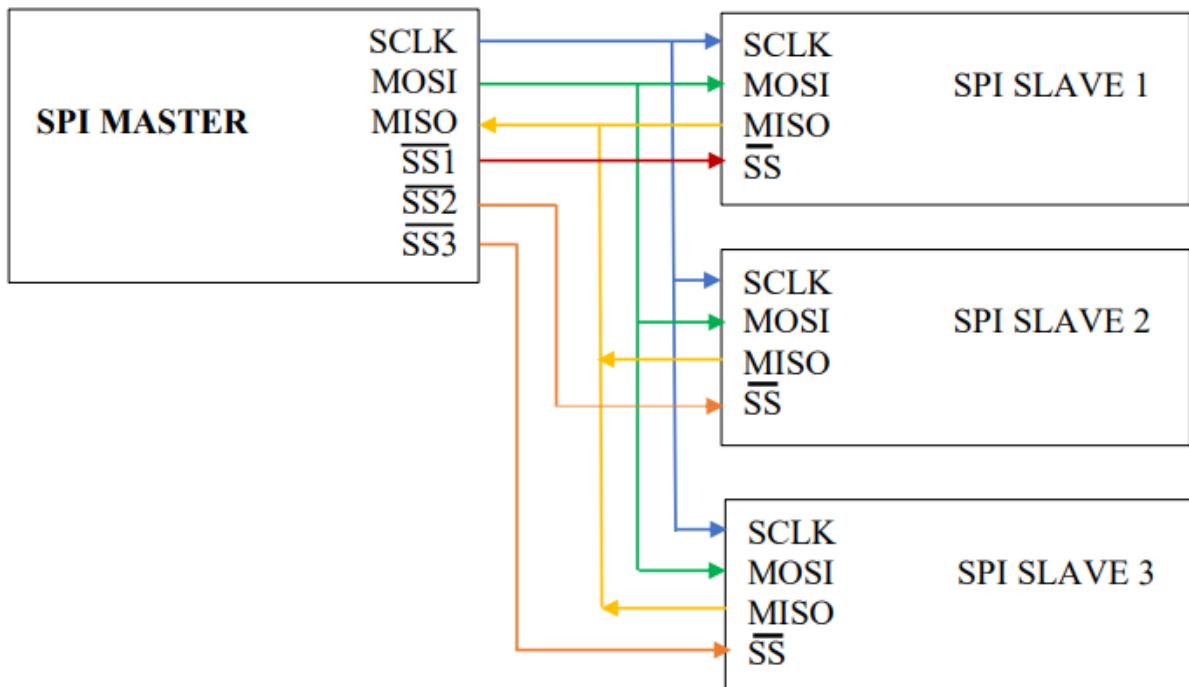
- תוך כדי שליחה של כל בית מה- SLAVE גם ה- MASTER מוציא בית אחריו בית נתון אל קו ה- MISO.
- גם הביטים מה- SLAVE מסונכרנים בעזרת פולסי השעון ב- SCLK והם נכנסים לרגיסטר הזהה ב- MASTER.
- בסיום שליחת הביטים ה- MASTER מעלה את קו בחירת רכיב עבד (Select slave) ל- 1 ומסיים תקשורת.
- בסיום התקשרות ה- MASTER לא מייצר פולסי שעון, והקו של פולסי השעון SCLK נמצא במצב IDLE מצב סרף. הוא יכול להיות ב- 0 או ב- 1.

### **חיבור מספר SLAVES אל ה- MASTER**

ניתן לחבר ל- SLAVES אחד מספר MASTER, מה שיקרא למעשה שהקווים המשותפים לכל ה- SLAVES הם SLCK, MISO, MOSI. קו שנבדל בין כולם הוא הקו לבחירת רכיב העבד (Slave select).

כל ה- SLAVES מקבלים במקביל את פולסי השעון ואת קווי ה- MOSI ו- MISO. קו ה- Slave select כל רכיב יהיה ב- 1 (כלומר הרכיב לא נבחר) והוא MASTER יוריד את קו בחירת רכיב העבד ל- 0 רק עבור אותו ה- SLAVE שהוא רוצה לתקשר אליו.

נראה את האיור הבא שמחיש לנו חיבור של 3 SLAVES ל- MASTER :



חיבור של MATER אחד לשולש SLAVES שונים דרך תקשורת טורית SPI.

באיור זה נוכל לראות שלכל עבד יש קו (Slave select) מסוילו. ל- MASTER יש אותו מספר קוויים בדומה למספר ה- SLAVES. לכל רכיב עבד יתחבר קו מסוילו אחר, באյור הם נקראים :

**(Slave select 1, Slave select 2, Slave select 3)**

הס מסומנים עם קו מעליים כדי להציגים שהם פעילים בנמוך (Active Low). דוגמה אם ה- MASTER יחליט לתקשר רק עם רכיב 1 SPI SLAVE הוא יוריד את קו (Slave select 1) ל- 0 ואז רק רכיב העבד הראשון ידע שהוא MASTER מתקשר אליו. שאר העבדים יודעים שהמייקרו בקר אינו מתקשר אליהם.

כמוות ה- SLAVES שנitinן לחבר אל ה- MASTER תלויים בכמות הבדיקות הפנויים שיש לו במערכת בה הוא נמצא ובקשר של דחיפה הזרם שלו. אפשרויות מתקדמות יותר על מנת לחסוך בחדקים של המיקרו בקר הון להתחבר אל הדקי Slave select של כל רכיב SPI בעזרת מנגנון או מפלג DEMUX עם כניסה אחת, 3 הדקי בחרה ו- 8 יציאות שונות.

#### **אפשרויות העבודה עם תקשורת SPI:**

במערכת ה- SPI (גם ה- MASTER וגם ה- SLAVE) יש 2 רגיסטרים, האחד הוא הרגיסטר הזזה המקבל את הדגימות ומזיז את הנטוון, עוד רגיסטר נתון שבסיום העברת הנטוון שהתקבל נמצא בו.

בՓולס שעון יש 2 מעברים (מגבוהה לנמוך ולהפך) הכוללים גם עלייה וגם ירידה ויש לעשות 2 דברים :

- .א. במעבר מ-0 ל-1 גם המסטר ו גם העבד מוצאים את בית הנטוון לקו הנטוון (I<sub>S</sub> MOSI בMASTER ו I<sub>S</sub> MISO בעבד).
- .ב. במעבר השני מתבצעת הזזה של הנטוון ברגיסטר ההזזה שנמצא גם בMASTER וגם בעבד.

#### **שני פרמטרים/מאפיינים חשובים בתקשרות SPI:**

- .1 CPOL – קוטביות השעון (Clock POLarity) – הקובעת מה מצב הקו (נמוך או גבוה) במצב סריך – IDLE כאשר אין פולסי שעון (לפני התחלת תקשורת ובסופה), כאשר CPOL=0 בקו יש 0 לוגי וכשהר CPOL=1 בקו יש 1 לוגי.
- .2 CPHA – Clock PHAse (פазת השעון) – הקובעת באיזה מעבר (האם מ-0 ל-1 או מ-1 ל-0) יוצא בית הנטוון בקו ה- MOSI ובקו ה- MISO ובאיזה מעבר הוא מוזז ברגיסטר הזזה גם בMASTER וגם בעבד. פרמטר נוסף חשוב הוא באיזה מצב נמצאו קו השעון SCLK בסיום התקשורת (האם ב 0 או ב 1 ).

#### **מבנה העברת של נתון עבור 0=CPHA:**

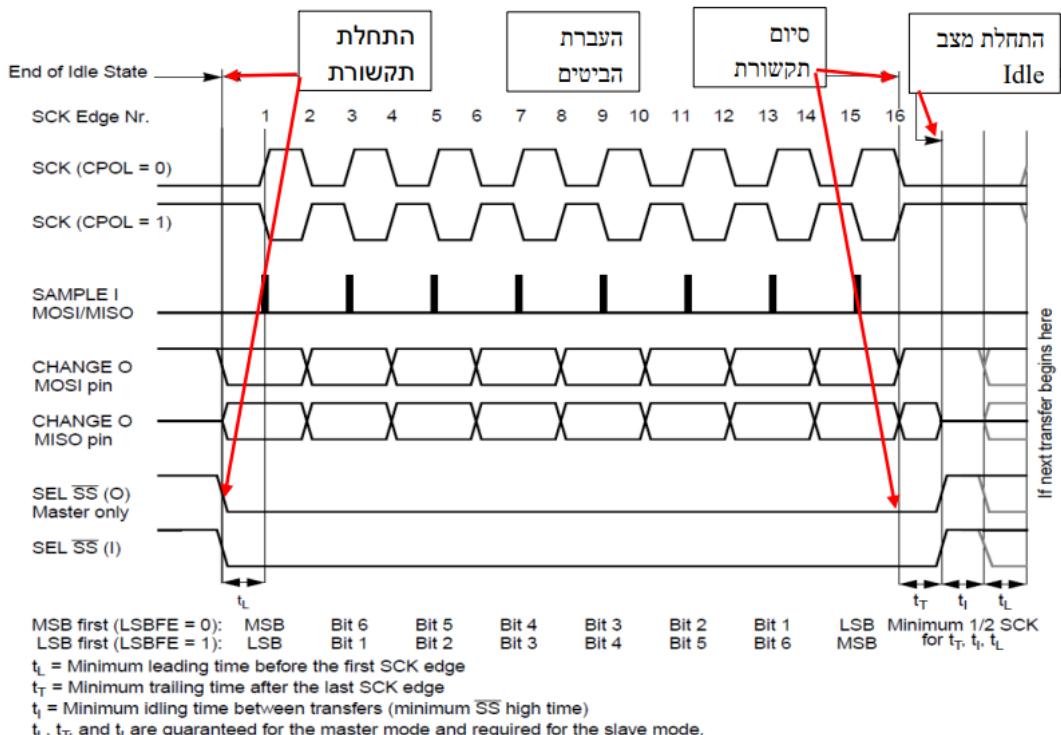
הקו Slave select פעיל בנמוך, ניתן לרשום אותו גם כך : NSS כדי שלא יהיה צורך לרשום גם SS. האירור הבא מתאר את תבנית העברת עם צורות הגל של תקשורת SPI כאשר CPHA=0 ועם פרמטרים של זמן אופיניים.

בחלק התחתיו רואים את התחלת התקשורת כאשר קו NSS (Slave select) יורד ל-0 ואת סיומו כאשר קו זה עולה ל-1.

קו השעון SCK מתואר על ידי 2 צורות גלים . העליונה ביותר מתארת את פולסי השעון כאשר CPOL = 0 (כאשר אין תקשורת יש 0 בקו פולסי השעון) והפולס הראשון הוא עלייה וצורת הגל שמתחתייה כאשר CPOL = 1 וזו יש 1 לוגי בקו פולסי השעון והפולס הראשון הוא ירידה. הראשון הוא ירידה.

במקומות הרושים באירור O הכוונה ל- Output (יציאה) ובמקומות שרשום I הכוונה ל- Input (כניסה). הביטים של הנטוון הטורי בהדקם MOSI ו- MISO נדגמים ומתבצעת הזזה ברגיסטרים של ההזזה באחת מ- 2 אפשרויות. או בעליית השעון או בירידת פולסי השעון ואת זה נקבע לפי נתוני הרכיב שאליוו מתחברים.

איור: צורת גל בתקשות SPI כאשר  $CPHA = 0$ .



המעבר הראשון של קו SCK (מ-0 ל-1 או מ-1 ל-0 תלוי ב-CPOL) משמש להכנסת בית הנטוון הראשון של העבד אל המスター (בקו ה-MISO) ואת בית הנטוון הראשון של המスター אל העבד (בקו ה-MOSI). באירור ניתן לראותו במרכזו את הקו שנקרא SEMPLE והוא משמש ככניסה גם למスター בקו ה-MISO וגם לעבד בקו ה-MOSI.

במעבר השני, בחצי המחוור הבא, מופיע מעבר נוסף בקו SCK ואוז הערך שנಡגס מהעבד בקו MISO נכנס לתוך רגיסטר ההזזה שבMASTER. אחרי המעבר הזה משודר הביט הבא של המスター על קו ה-MOSI התהיליך נמשך עד 16 מעברים בקו SCK כאשר נטען **נעל אל המスター במעברים איזוגיים ומועבר אל העבד במעברים זוגיים**.

אחרי מעבר מס' 16 הנטוון יהיה ברגיסטר ה-SPI של המスター צריך להיות ברגיסטר הנטוון של העבד והנתוון יהיה ברגיסטר הנטוון של העבד צריך להיות בMASTER.

#### באיור מופיעים זמינים בגוון:

- $t_L$  המראה מה הזמן המינימאלי מרגע הורדת קו NSS (Slave select) ל-0 ועד להתחלה פולסי השעון.
- $t_T$  הוא הזמן המינימאלי בין פולס השעון האחרון והעלאת קו NSS (Slave select) ל-0.
- $T_i$  הוא זמן הסרק-IDLE – בין העלאת קו NSS (Slave select) ל-1 ועד שידור נתון חדש על ידי הורדת NSS (Slave select) ל-0.

מתוחת לצורות הגלים ניתן לראות את הנטוון מביט ה-LSB אל ה-MSB או להיפך.

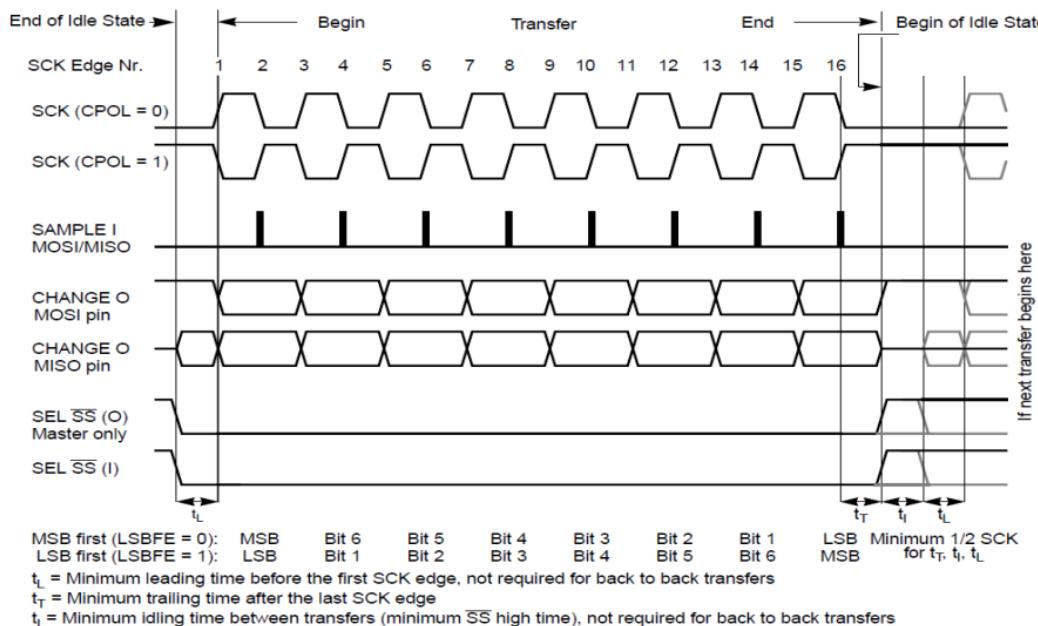
קיימים ביט LSB First Enable (LSBFE) אשר LSB First Enable (LSBFE) שקבע מי נכנס ראשון. כאשר הוא 0 נכנס ביט ה-MSB ראשון וכאשר הוא 1 נכנס ה-LSB ראשון.

קיבולת הנטוון מתבצעת בשני שלבים. היא מזוזת לרגיסטר ההזזה של ה-SPI בזמן ההעברה וMOVEBERT לרוגיסטר הנטוון של ה-SPI כאשר הבית האחרון הווז פנימה.

### תבנית העברת של נתון עבור 1 : CHPA

קייםים רכיבים שצרכיהם את המעבר של פולס השעון הראשון לפני שניתן יהיה לגשת לביט הנתון הראשון בכו היציאה של הנתון. המעבר השני מכניס את הנתון למערכת. פורמט זה מתקבל על ידי השמה של A-L-CPHA.

#### האיור הבא מ\_tAר את התקשורות עבור 1 .CPHA =

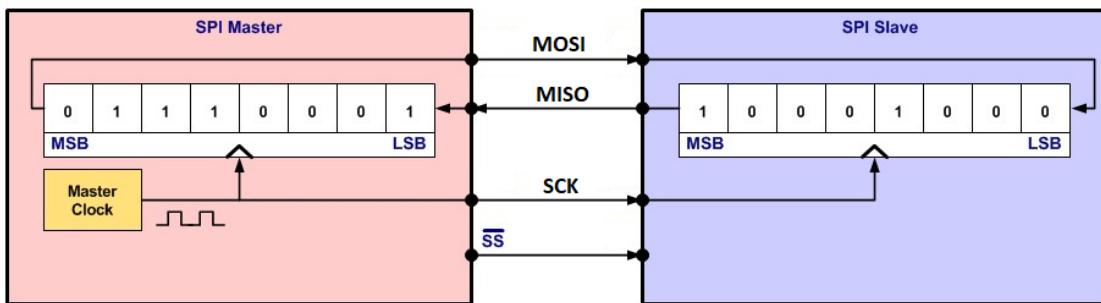


באיור רואים שהמעבר הראשון משמש כהשטיית סנכרון ואומר לעבוד להוציא נתון בכו ה- MISO. בחצי המחוור הבא, במעבר השני, יש געילה של בית הנתון גם בMASTER וגם בעבד. כאשר מגיע המעבר השלישי מועבר הביט שנגע במעבר השני לתוך ה- MSB או ה- LSB של רגיסטר החזזה (כתלות בבית LSBFE).

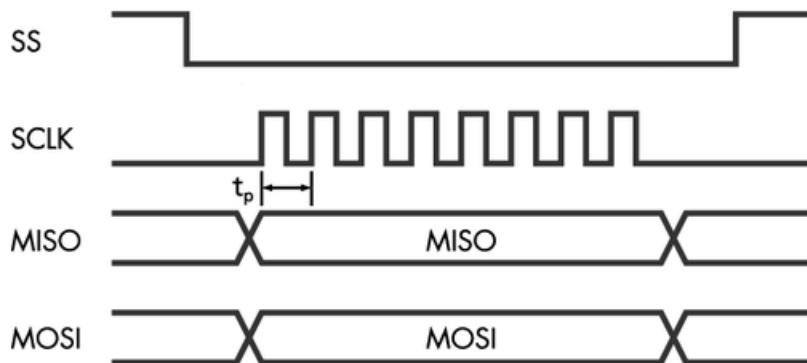
אחרי מעבר זה יוצא בית הנתון הבא של המסטר בכו ה- MOSI. התהליך חוזר על עצמו עבור כל 16 המעברים, כאשר הנתון נגע במעברים הזוגיים וההזזה קוראת במעברים האי זוגיים.

המידע מועבר בו זמינות בשני הכוונים בכל עליית שעון באמצעות אוגרי הזזה .

**בדוגמה הבאה:** לאחר 8 מחזורי שעון מידע מוחלף בין ה- Master ל- Slave .



שליחת מילת בקרה של 8 סיביות.



בתחילת התקשרות, הדק SS יורדת ל- 0 . לאחר מכן מועבר המידע מה- Master דרך הדק מה- Slave . Master יכול לקרוא את המידע מה- Slave דרך הדק MISO . בסיום התקשרות הדק SS חוזרת ל- 1 lagi .

$t_p$  – זמן מחזור של השעון .

## תקשורת I2C

### **הקדמה:**

פרוטוקול I2C הוא שיטת תקשורת סריאלית נפוצה בין התקני אלקטרונייקה וחישנים. הוא מאפשר העברת נתונים בין מספר התקנים, כאשר יתרונו הגדול הוא שהוא דורש רק שני חוטים לתקשורת – החוט לאותות השעון והחוט לאות הנתונים.

### **מהו פרוטוקול I2C:**

פרוטוקול I2C (אי-שטיים-סי) הוא שיטה סריאלית להעברת נתונים בין התקנים, שפותחה על ידי חברת פיליפס בשנת 1980.

בניגוד לתקשורת UART שדורשת שני חוטי תקשורת לכל כיוון, ב-I2C יש רק שני חוטים משותפים לשני הכוונים: SDA לנtones ו-SCL לאות שעון. שני חוטים אלו מחוברים לכל החתקנים בטופולוגיה של BUS.

כל התקן ברשות I2C מזוהה בכמות ייחודית. כאשר מתokin אחד רוצה לתקשר, הוא שולח הודעה עם הכמות של המתקן היעד. רק המתקן עם אותה כמות "יורם את השפופרת".

העברת הנתונים עצמה נעשית לפי עקרון מסטר/עבד (לרוב מסטר עובד), מפעיל את אות השעון ושולט מתי לשדר ומתי לקבל נתונים, ואילו המתקנים המשניים הם עובדים פסיביים.

ב-I2C כל בית נשלח ברכז אחורי הבית הקודם, והוא תוקף כל עוד אות השעון SCL גבוהה. ירידת SCL מהווה אינטואיטיבית לבית חדש.

יתרונותיו הגדולים של פרוטוקול I2C הן הפשטות והחיסכון בכמות החוטים לתקשורת, ניתן לחבר עשרות מתקנים קצה, וכל יחסית ליישם במיקרו-בקרים.

לכן זהו פרוטוקול נפוץ לתקשורת עם אלקטרונייקה כמו חישנים, מסכי LCD, גרפים, זיכרונות וכו'.

### **מאפיינים פרוטוקול I2C:**

- **תקשורת סיידרטייט (סריאלית)** - המידע מעובר בטור, בית אחורי בית, על גבי קו התקשורת SDA.
- **סינכרוניות** - אות השעון SCL מסנכרון בין כל המכשירים.
- **תקשורת דו כיוונית** בין מסטר למספר סלייבים.
- **שני קווי תקשורת** משותפים לכל המכשירים SDA ו-SCL.
- **קצב תקשורת** גבוה יחסית - עד 5 מגה ו אף יותר.
- **כבות מוגדרות** מראש לכל רכיב (10/7 בית כמות).
- **ממשק חומרה פשוט וסטנדרטי.**
- **מתאים במיוחד** לתקשורת בין מעבד להתקני קצה כמו חישנים, מסכיים ו זיכרונות.
- **מצמצם מאוד** בכמות החיווט לעומת UART.

### מבנה חיבור (מסגרת) נתונים בתקשורת טורית:

כאשר מתקן מסטר (כמו מיקרו-בקר) מעוניין לשדר נתונים למתקן SLAVE, הוא בונה את החיבור במבנה מוגדר הכלל מספר שדות:

- **התחלת העברה (START)** - ירידה באות SDA מ-1 ל-0 בזמן שאות SCL נשאר גבוה, מה שמהווה "אות התחלת".
- **כתובת SLAVE** - 7 או 10 ביטים עם הכתובת הייחודית של המתקן SLAVE היעד. רק אותו מתקן ייריב את השופורת.
- **נתונים** - המידע עצמו בסדרת ביטים הנשלחים על ידי המאסטר או שמתקבלים מהמתקן SLAVE.
- **אישור 9 (ACK)** - פעולות שעון שבוחן ה-SLAVE מאשר קיבל תקינה של העברת 8 ביטים בקו SDA.
- **סיום העברה (STOP)** - עלייה באות SDA מ-0 ל-1 בזמן שקו SCL נשאר גבוה, מה שמהווה "אות סיום".

### דוגמה: שליחת בית ב프וטוקול I2C:

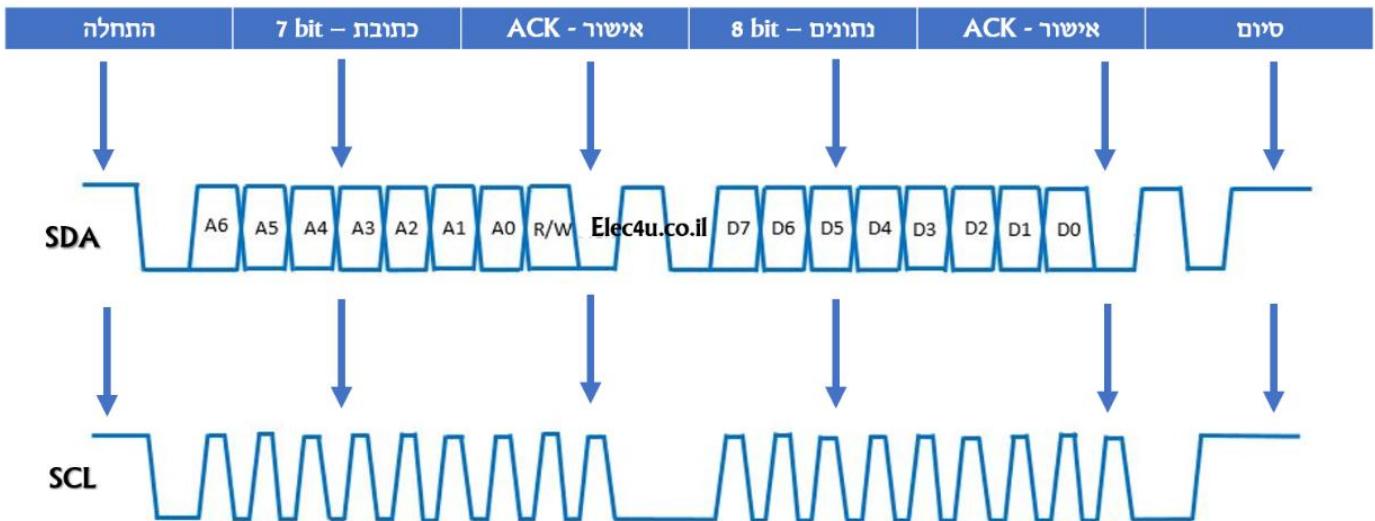
הנה פירוט דוגמה לשילוח הביטי  $0x2F$  (מספר 47 עשרוני) למתקן בכתובת  $0x38$  (מספר 56 עשרוני):

- $00101111 \rightarrow ACK \rightarrow STOP \rightarrow 0x38$  (כתובת)  $\rightarrow START \rightarrow 0111000$

### פירוט שלבי התקשורת:

- **START** : מצין על התחלת העברת חדשה. נוצר על ידי ירידה של קו הנתונים SDA מ-1 ל-0, בעוד קו השעון SCL נשאר גבוה.
- **0x38** (כתובת) : שבעת הביטים הבאים מצינים את כתובת ה-SLAVE (במקרה זה 47 עשרוני). הם נשלחים מהבית הנמוך (LSB) לביט הגבוה ביותר (MSB).
- **00101111** : שבעת הביטים הבאים מצינים את הנתונים שאנו שולחים ל-SLAVE (במקרה זה 56 עשרוני או  $0x2F$ ). שוב, מ-LSB ל-MSB.
- **ACK** : לאחר הנתונים, ה-SLAVE חייב לשЛОוח בית "אישור" אחד בחזרה לMASter כדי מאשר שקלט את הנתונים כראוי.
- **STOP** : סיום תנועת ה-I2C. נעשה על ידי העלאת קו הנתונים SDA בחזרה מ-0 ל-1, בעוד קו השעון SCL נשאר גבוה.

### נתבונן באיור הבא:



בחבילת נתונים של פרוטוקול I<sub>2</sub>C ישנה אפשרות לשЛОח מסגרת אחת או מספר מסגרות ברכז באותה החבילה. מוגרת נתונים אחת מכילה את כמות המידע שאנו רוצים לשЛОח באותו הרגע. הפרוטוקול מאפשר לנו לשLOB מספר מסגרות נתונים ברכז בתוך אותה החבילה, ללא צורך לשLOW חדשה עBOR כל מסגרת. זאת על ידי שליחת אישור (ACK) מהתקן מקבל בין מסגרת למסגרת. כך ניתן לשLOW ביעילות נתונים ארוכים יותר באותו החבילה I<sub>2</sub>C.

### יתרונות מרכזיים של פרוטוקול I<sub>2</sub>C :

- **חסכוני בכמות החיווט** - משתמשים רק בשני קווי תקשורת משותפים לכל התקנים - קו SDA לנواتים וקו SCL לאוות השעון.
- **תקשורת דו-כיוונית** - מאפשר הרחבות למאסטר לשLOW נתונים והן ל-SLAVE להחזיר נתונים.
- **פשוט לחיבור התקני קצה רבים** - ניתן לחבר עשרות התקנים שונים על אותה רשת I<sub>2</sub>C.
- **ממשק תקשורת פשוט וסטנדרטי** - כל רכיבי החומרה מתוכננים לתמוך בפרוטוקול.
- **מהירות תקשורת גבוהה** - עד 5 מגה-הרץ, מותאים להעברת נתונים מחישנים, שליטה ועוד.
- **אמינות גבוהה** - שימוש ב-ACK-ומנגנון זיהוי שגיאות.
- **נתיבות מוגדרות מראש** - מאפשר תקשורת מכוונת בין התקנים הרשא.

### כמויות רכיבים שנייתן לחבר לפרטוקול I2C:

בפרטוקול I2C ניתן לחבר מספר רב של **רכיבים**, אך ישנה הגבלה על הכמות המקסימלית. בפועל, ניתן לחבר עד 128 **רכיבי קצה (SLAVES)** לכל רשות I2C. הסיבה להגבלת 128 היא **שימוש ב-7 ביטים לייצוג כתובת הריבב בפרטוקול**, מה שמאפשר טווח כתובות של 0 עד 127 (כלומר 128 כתובות שונות). כמות זו של 128 רכיבים נחשבת כבואה יחסית ומספריקה לרוב המערכות והיישומים הנוכחיים.

### הרחבה ל-10 ביטים:

ישנה אפשרות להשתמש גם בכתובות עם 10 ביטים, מה שמרחיב את טווח הכתובות ומאפשר חיבור של מספר רב יותר של רכיבים. עם זאת, השימוש ב-10 ביטים מורכב יותר ופחות נפוץ.

### הגבלות נוספות:

מלבד מגבלת הכתובות, קיימות גם **מגבליות פיזיות** שאלוולות להשפעה על כמות הרכיבים המקסימלית שנייתן לחבר בפועל. מגבלות אלו כוללות:

- **קיבוליות (Capacitance)**: ככל שמוסיפים יותר רכיבים, כך גדלה הקיבוליות על קווי התקשורת SCL ו-SDA. קיבוליות גבוהה מדי יכולה לעוות את האותות החשמליים ולשבש את התקורת.
- **אורך החוטים**: ככל שהחוטים ארוכים יותר, כך גדלים הסיכון לרעש חשמלי ולירידת מתח.
- **זיהום סביבתי**: סבيبة רועשת מבינה חשמלית עלולה להשפיע על האותות.

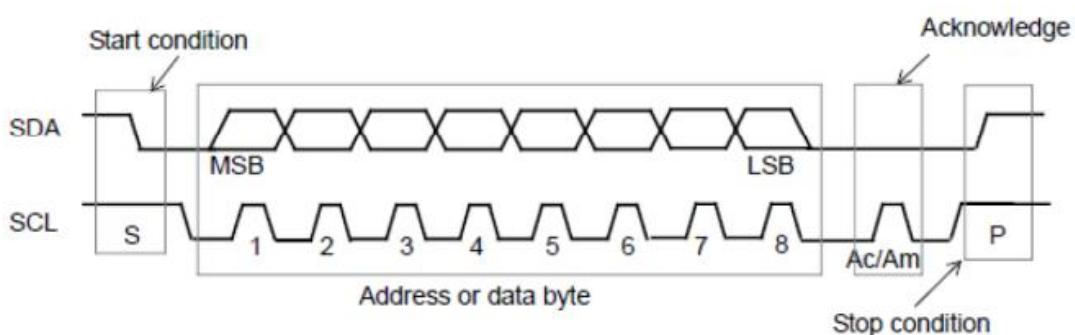
בפועל, למروת התיאוריה של 128 כתובות, יש לנקוט בחשבון את המגבליות הללו, ולרוב רשותות I2C פשוטות מוגבלות למספר קטן יותר של רכיבים.

### פעולות בסיסיות בפרטוקול התקורת I2C:

כאשר שני האותות SDA ו- SCL, נמצאים במצב 1 לוגי ה- BUS במנוחה. השידור עצמו נעשה בשיטת MSB – First.

- האות SDA חייב להיות יציב כשהאות SCL במצב 1 לוגי.
- האות SDA יכול להשתנות רק כשהאות SCL במצב 0 לוגי.

**נראה תמונה שמתארת העברת מידע בפרטוקול בדומה לתמונה מהעמוד הקודם:**



ניתן לראות שפעולות START מתבצעת כאשר SDA בירידת ו- SCL נמצא ב-1 לוגי. לעומת זאת פעולת STOP מתבצעת כאשר SDA בעלייה ו-SCL כבר ב-1 לוגי.

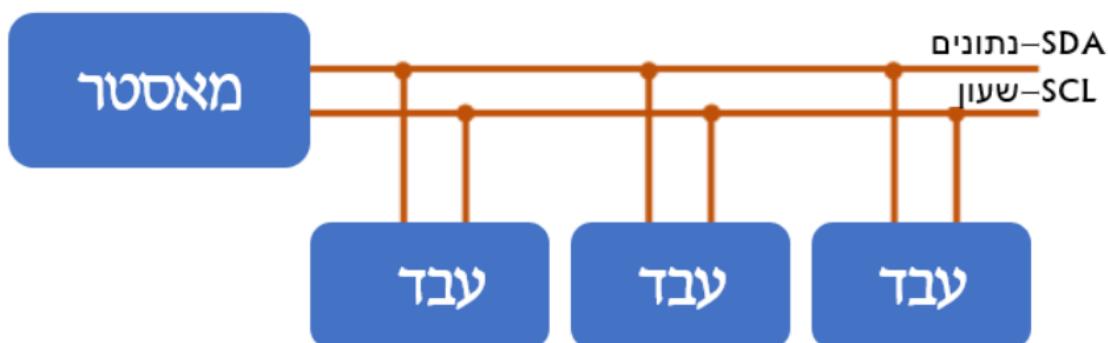
הרכיב שמקבל את המידע (ה-MASTER) או ה-SLAVE מקבלים Acknowledge (SLAVE) מחזקירים (MASTER) תמיד להשיב באוטו Acknowledge לפניהו לכתובות שליהם.

מצבים בהם רכיבי SLAVE לא מגיבים ב- Acknowledge :  
רכיבי SLAVE יכולים שלא להגיב בתגובה Acknowledge (כלומר להגיב ב- NACK), במקרים הבאים :

- ה-MASTER עסוק (במקרה כזה הוא יכול גם להאריך את ה-SCL).
- ה-SLAVE קיבל מידע לא חוקי.

לאחר קבלת NACK ה-MASTER חייב לבצע פעולה STOP. כאשר ה-MASTER הוא ה-Receiver הוא יכול שלא להגיב באוטו Acknowledge (כלומר להגיב ב- NACK) כדי לסמן שהגיע ל- Byte הסופי של המידע. מערכות שפועלות ב프וטוקול I2C יכולות לבצע פעולה Byte Acknowledge גם ברמת ה- Byte.

#### הסביר כיצד ה프וטוקול I2C פועל:



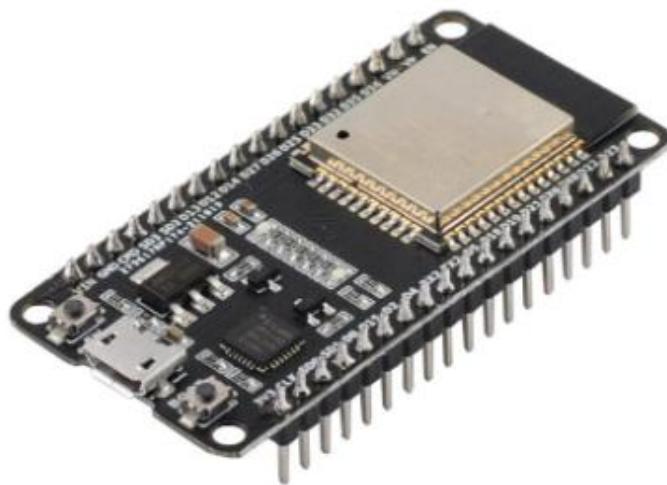
פרוטוקול I2C הוא שיטת תקשורת טורית בין מיקרו-בקר לרכיבי ציוד היקפי, המשמשת בשני קווים בלבד.

- רgel - (SDA) Serial Data אחראית על שליחת וקבלת המידע בין המכניםים.
- רgel - (SCL) Serial Clock אחראית על אותן השעון, המسانדרן את התקשרות בין כל רכיב, כך שהם "מדברים" יחד ובזמן הנכון.

הארדואינו/ESP32, הם ה"מאסטרים" והוא שולט על הקווים, והשאר הם ה"עבדים" (SLAVES). כאשר הארדואינו/ESP32 רוצה לתקשר עם אחד החישנים, הוא שולח אליו את הכתובת הייחודית שלו. לדוגמה :

- תצוגת 16x2 LCD I2C עובדת בכתובת 0x27.
- חיישן טמפרטורה LM75 עובד בכתובת 0x07.

## מיקרו בקר ESP32



ה- **ESP32** הוא הרבה יותר מסתם מיקרו-בקר פשוט ; הוא מערכת על שבב (SoC) חזקה ורב-תכנית, המשלבת את כל מה שצריך כדי לבנות פרויקט חכם, מוקשר וחסכוני באנרגיה. פרי-פיתוחה של חברת **Espressif Systems**, ESP32 בולט בזכות השימוש הייחודי שלו עצמת עיבוד חזקה, קישוריות אלחוטית וכיוכלה קריכת חשמל נמוכה.

### תכונות טכניות עיקריות:

- .**1. עיבוד בעל ביצועים גבוהים** : מצויד בمعالג כפול-ЛИבָה במהירות של עד 240 MHz, ה- ESP32 מסוגל לבצע משימות מורכבות ולנהל מספר פעולות במקביל במהירות גבוהה.
- .**2. קישוריות אלחוטית מובנית** : ה- ESP32 מגיע עם **Wi-Fi** ו- **Bluetooth** מובנים, מה שהופך אותו לבחירה אידיאלית לפיתוח פרויקטים מחוברים ללא צורך ברכיבים חיצוניים נוספים.
- .**3. יעילות אנרגטית** : עם מצבי שינה מרובים, ה- ESP32 מתוכנן לפעול לטוווח ארוך על סוללה בודדת, מה שהופך אותו למושלם עבור יישומים המופעלים באמצעות סוללה.
- .**4. ממשקים קלט/פלט (I/O) גמישים** : ה- ESP32 מציע מגוון רחב של ממשקים היקפיים, כולל **GPIO, ADC, DAC, I2C, SPI**.
- .**5. תמיכת קהילה נרחבת** : עם אלפי מפתחים ברחבי העולם המשתמשים ב- ESP32, קיימים שפע של ספריות קוד, תיעוד ופרויקטי קוד פתוח הזמינים לתמיכת בפיתוח שלך.

### **הסוד שמאחורי ה-ESP32: יחידת העיבוד המרכזית (CPU)**

ה-ESP32 מצויד בمعالג עצמאי Xtensa LX6 dual-core, הפועל במהירות מרשימה של עד 240MHz. אבל מה זה בעצם אומר?

- דמיינו שיש לנו שני מוחים עובדים במקביל: בזמן אחד מהם מטפל בקשריות-Wi-Fi, השני יכול לעבוד נתונים מהחישונים שלכם - הכל בבת אחת!
- **240MHz** זהה למהירות השעון של המعالج. לשם השוואה, המعالج של ה-ESP32 מסוגל **לבצע 240 מיליון פעולות בשנייה!**
- **ארQUITקטורת 32-ביט** זה אומר שהוא יוכל לעבוד נתונים בכמותות גדולות יותר, מה שמאפשר חישובים מורכבים במהירות גבוהה.

### **הזיכרון: איפה כל המידע נשמר?**

ה-ESP32 מגיע עם מערך מרשים של אפשרויות זיכרון:

- **SRAM (Static Random-Access Memory)**: זיכרון מהיר לשימוש מיידי של 520KB. זהו המקום שבו המידע שהمعالגעובד עליו ברגע נתון נשמר.
- **ROM (Read-Only Memory)**: זיכרון קבוע של 448KB המכיל את ההוראות הבסיסיות הנדרשות להפעלת המערכת.
- **Flash (זיכרון חיצוני)**: זיכרון של עד 16MB שבו נשמרות תוכניות המשמשותם והנתונים הקבועים.

לשם השוואה, הזיכרון הכלול של ה-ESP32 הוא כל כך גדול, שניתן לאחסן בו ספר אודיו שלם!

### **קשריות אלחוטית: החיבור לעולם:**

ה-ESP32 הוא אלף התקשורות האלחוטית:

- **Wi-Fi**: תומך בטקן 802.11 n/g/b, מה שמאפשר חיבור כמעט ללא כל רשות אלחוטית מודרנית. הוא יכול לשמש גם כנקודת גישה ולספק חיבורם למכשירים אחרים.
- **Bluetooth Low Energy (BLE)**: עם תמיכה ב-Bluetooth Classic וגם ב- BLE, ה-ESP32 יכול לתקשר עם מגוון רחב של מכשירים, טלפונים חכמים ועוד חישונים צעירים.

### **ממשקים: חיבור לעולם הפיזי:**

ה-ESP32 מציע מגוון רחב של אפשרויות חיבור כדי לתקשר עם העולם הפיזי:

- **GPIO (General Purpose Input/Output)**: עד 36 פינים, המשמשים כ"עצבים" של ה-ESP32, המאפשרים לו לחוש ולשנות בסביבה.
- **ממיר אנלוגי-דיגיטלי (ADC)**: ישנו 18 ערוצים המאפשרים ל-ESP32 ל לקרוא ערכים אנלוגיים מחישונים כמו טמפרטורה או עצמת אור.
- **ממיר דיגיטלי-אנלוגי (DAC)**: שני ערוצים המאפשרים לייצר אותן אוטות אנלוגיים כמו צלילים, או מתחים משתנים.
- **חישוני מגע (Touch Sensors)**: 10 פינים מגע מובנים המאפשרים ל-ESP32 לחוש מגע ישיר.

### צריכת חשמל:יעילות אנרגטית מרשימה:

אחד ה יתרונות הגודלים ביותר של ה-ESP32 הוא יכולת שלו לחסוך באנרגיה :

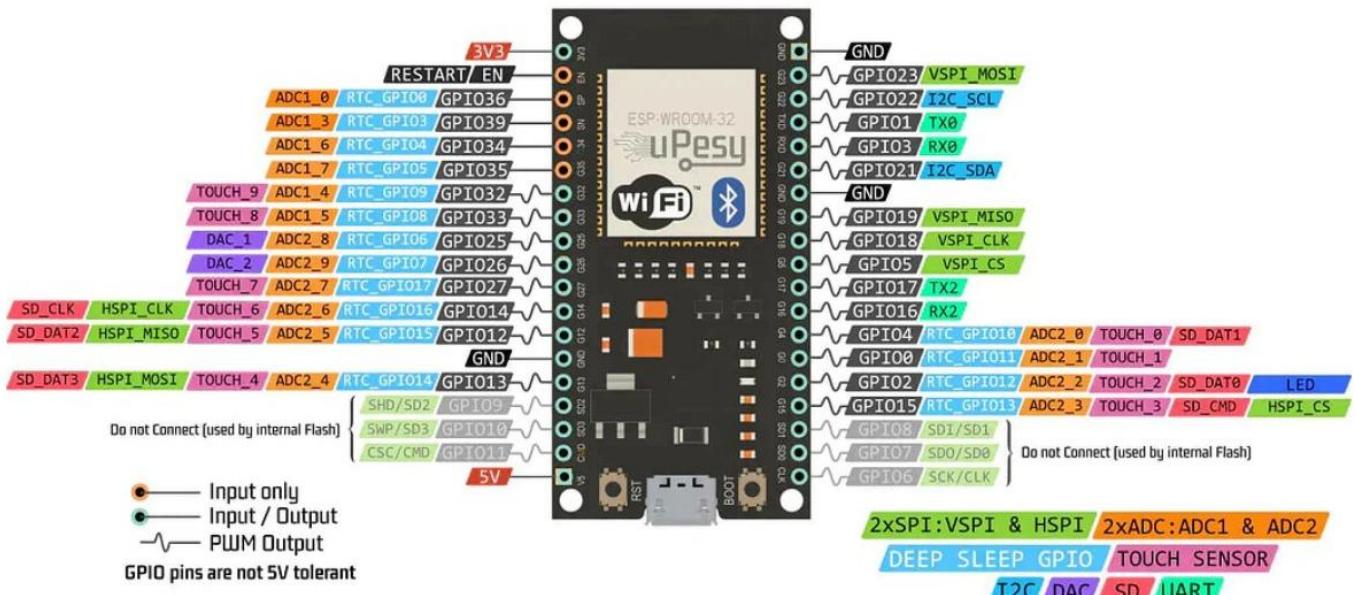
- **מצב פעיל מלא:** במצב זה, צריכת הזרם היא עד **.mA240**.

- **מצב שינה عمוקה:** במצב זה, הצריכה יורדת באופן דרמטי ל-**A10μ** בלבד!

זה אומר שגם עם סוללה קטנה, הפרויקט שלכם יוכל לפעול במשך שבועות ואף חודשים ארוכים.

### כעת נראה את צורת הפנים במקרו בקר :ESP32

#### **ESP32 Wroom DevKit Full Pinout**



כפי שכבר רأינו, ה-ESP32 מוקף בפינים רבים, שהם ה"שערים" דרכם הוא מתקשר עם העולם החיצוני. הבנה מעמיקה של פינים אלו היא המפתח לייצור פרויקטים מודרכים. בואו נצלול לעולם המרתך של פיני ה-ESP32 :

### סוגי הפינים העיקריים:

#### **פיני (General Purpose Input/Output)**

- אלו הם הפינים הרוב-תכלתיים של ה-ESP32.
- יכולים לשמש כקלט (לקראת מצבים) או כפלט (להפעלת רכיבים).
- ה-ESP32 מציע עד **34 פיני GPIO**, אך חלקם משמשים גם למטרות אחרות.
- **שימושים נפוצים:** חיבור נורות LED, לחצנים, מנועים ועוד.

### :ADC (Analog to Digital Converter)

- אפשרים קרייה של מערכות אנלוגיות ומרתם למערכות דיגיטליות.
- ה-ESP32 מציע שני מגלי ADC עם סה"כ **18 ערוצים**.
- רזולוציה של **12 ביט** מאפשרת **4,096 רמות שונות** של קרייה.
- שימושים:** מדידת טמפרטורה, עצמת אור, לחות קרקע ועוד.

### **:DAC (Digital to Analog Converter)**

- ממירים ערcis דיגיטליים לאוותות אנלוגיות.
- ה-ESP32 מציע שני ערוציו DAC ברזולוציה של **8 ביט**.
- **שימושים נפוצים:** יצירת צלילים, בקרת מתח ועוד.

### **:Touch**

- ה-ESP32 מציע **10** חיישני מגע קפסיטיביים מובנים.
- אפשררים זיהוי מגע אנושי ישיר, ללא צורך ברכיבים נוספים.
- **שימושים נפוצים:** יצירת ממשקי משתמש מבוססי מגע, לחצנים וירטואליים ועוד.

### **:פיני תקשורת:**

- **UART:** לתקשורת טורית. ה-ESP32 מציע **3 יחידות UART**.
- **SPI:** לתקשורת מהירה. ה-ESP32 תומך במספר חיבורו SPI במקביל.
- **I2C:** לתקשורת עם מגוון רחב של חיישנים ורכיבים. ה-ESP32 מציע שתי יחידות **I2C**.

### **:פינים מיוחדים:**

- **פין BOOT:** משמש לכינסה למצב **כרייבת תוכנה (flashing mode)**.
- **פין (EN) (Enable):** משמשים להפעלה וכיבוי של השבב.
- **פיני אספקת מתח:** פינים המספקים מתחים של **5V, 3.3V** וכן **GND**.

### **:נקודות חשובות לגבי השימוש בפינים:**

- **רמות מתח:** פיני ה-ESP32 עובדים ברמת מתח של **3.3V**. חיבור למתח גבוה יותר עלול לגרום נזק.
- **פינים מרובי תפקידים:** רבים מהפינים יכולים לשמש במספר מטרות. חשוב לתכנן את השימוש בהם בקפידה.
- **הגנה על הפינים:** מומלץ להשתמש ב נגדים מגבילים זרם בעת חיבור רכיבים חיצוניים, במיוחד עם נורות LED.
- **פינים מיוחדים:** חלק מהפינים GPIO (כמו 0, 2, 15) משמשים במהלך תהליך האתחול. שימוש לא נכון בהם עלול למנוע מה-ESP32 לפעול כראוי.
- **פולאף ופולאדיון (Pull-up and Pull-down):** חלק מהפינים כוללים גודי Pull-Up או Pull-Down פנימיים. זה יכול להוות שימוש חשוב, אך חשוב להיות מודעים לכך בעת תכנון המעגל.

הבנה מעמיקה של מבת הפינים ותכונותיהם היא המפתח לייצרת פרויקטים מוצלחים ויעילים עם ESP32. זה מאפשר לנו לנצל את מלאה הפוטנציאלי של השבב העוצמתי הזה.

### **:הכוח האלחוטי של ESP32**

אחד ה יתרונות הגודלים ביותר של ה-ESP32 הוא יכולתו המובנית לתקשורת באמצעות אלחוטי.

## **החיבור לעולם הרחב Wi-Fi**

ה-ESP32 תומך בתקן **Wi-Fi 802.11 b/g/n**, מה שמאפשר לו להתחבר כמעט לכל רשת אלחוטית מודרנית.

### **מצבי פעולה עיקריים:**

#### **• מצב תחנה (Station Mode - STA)**

- ה-ESP32 מתחבר לרשת Wi-Fi קיימת, כמו כל מכשיר קצה רגיל.
- שימושים נפוצים: התחברות לאינטרנט דרך הרouter הביתי, שליחת נתונים למסד נתונים מרוחק ועוד.

#### **• מצב נקודת גישה (Access Point - AP):**

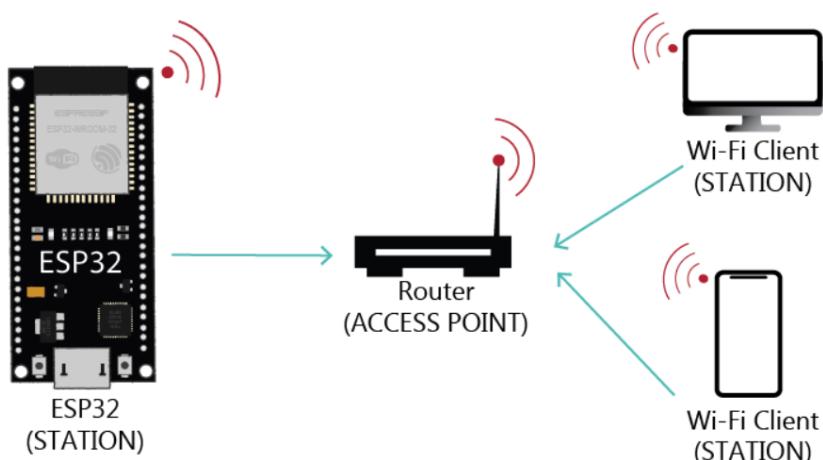
- ה-ESP32 יוצר רשת Wi-Fi משלו, אליה יכולים להתחבר מכשירים אחרים.
- מצוין לייצור פרויקטים שימושיים ממשק אינטרנט או תקשורת ישירה עם מכשירים ניידים.
- שימושים נפוצים במצבים בהם אין גישה לרשת Wi-Fi חיצונית.

#### **• מצב כפול (Dual Mode)**

- ה-ESP32 יכול לפעול במקביל גם כתחנה וגם כנקודת גישה בו-זמנית!
- אפשר ליצור "גשר" בין רשתות או אספקת גישה לאינטרנט דרך ה-ESP32.

## **יכולות מתקדמות של Wi-Fi:**

- **שירות אינטרנט (Web Server):** ה-ESP32 יכול לשמש כשרת אינטרנט, מה שמאפשר יצירה ממשקי משתמש מרוחק ודף תצוגה נתונים.
- **MQTT:** תמיינה ב프וטוקול זה מאפשרת תקשורת יעילה בין מכשיר IoT ושרותים מרכזיים.
- **OTA (Over-The-Air) Updates:** עדכון תוכנה מרוחק, ללא צורך בחיבור פיזי למחשב.
- **ESP-NOW:** פרוטוקול תקשורת ייחודי של Espressif המאפשר תקשורת ישירה בין מכשירי ESP32 בזריכת אנרגיה נמוכה.



### **: תקשורת קצרה טווח עם יכולות ארכוות טווח Bluetooth**

ה-ESP32 תומך הן ב-(BLE) Bluetooth Low Energy והן ב-(Classic), מה שפותח אפשרויות רבות:

#### **:Bluetooth Classic**

- מתאים לתקשורת עם מכשירים ישנים יותר או כנדרשת העברת נתונים בנפח גדול.
- שימושים נפוצים: הזרמת אודיו, העברת קבצים, ותקשורת עם מכשירים שאינם תומכים BLE.
- אפשר יצירה פרופילים שונים כמו **Serial Port Profile (SPP)** לתקשורת טורית אלחוטית.

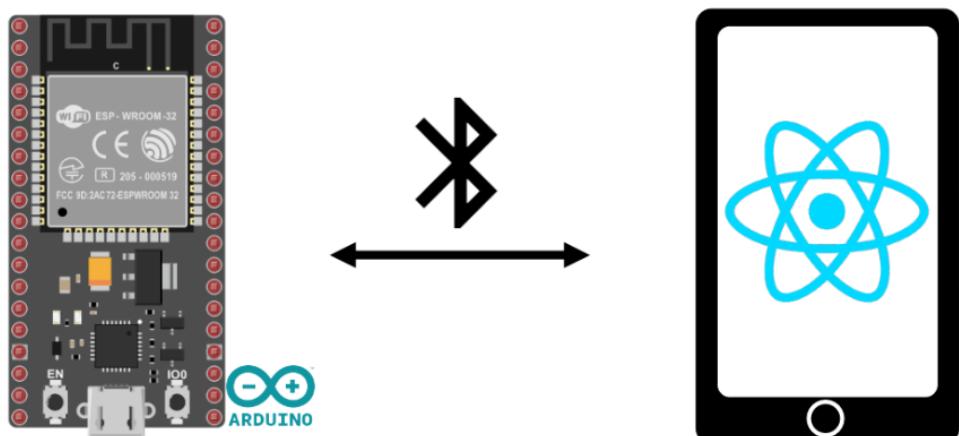
#### **:Bluetooth Low Energy (BLE)**

- צריכה אנרגיה נמוכה במיוחד, אידיאלי למכשירים המופעלים על סוללה.
- מתאים לשילוח נתונים קטנים באופן\_TDיר, כמו נתונים חיישנים.
- תומך במודול **GATT (Generic Attribute Profile)** המאפשר יצירה של שירותים ומאפיינים מותאמים אישית.
- אפשר יצירה "bijonics" לשיווק מיידי ללא צורך בחיבור מלא.

### **שילוב Wi-Fi ו-Bluetooth: יתרונות מרתקים:**

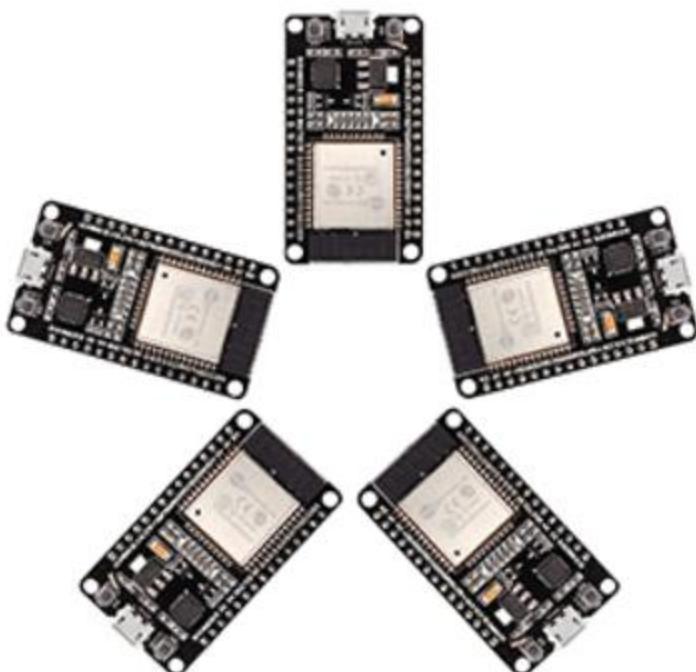
אחד היתרונות הגדולים של ה-ESP32 הוא יכולת להשתמש גם ב-Bluetooth וגם ב-Wi-Fi במקביל. זה פותח אפשרויות מרתקות:

- **שער (Gateway):** קבלת נתונים ממכשירי Bluetooth ושליחתם לענן דרך Wi-Fi.
- **שליטה מקומית ומרוחקת:** שליטה בפרויקט דרך Bluetooth כשלרבים, ודרך Wi-Fi כশמרוחקים.
- **רשתות היברידיות:** שימוש ב-BLE לתקשורת בין חיישנים קרובים, וב-Wi-Fi לשילוח נתונים רבים לשרת מרכזי.

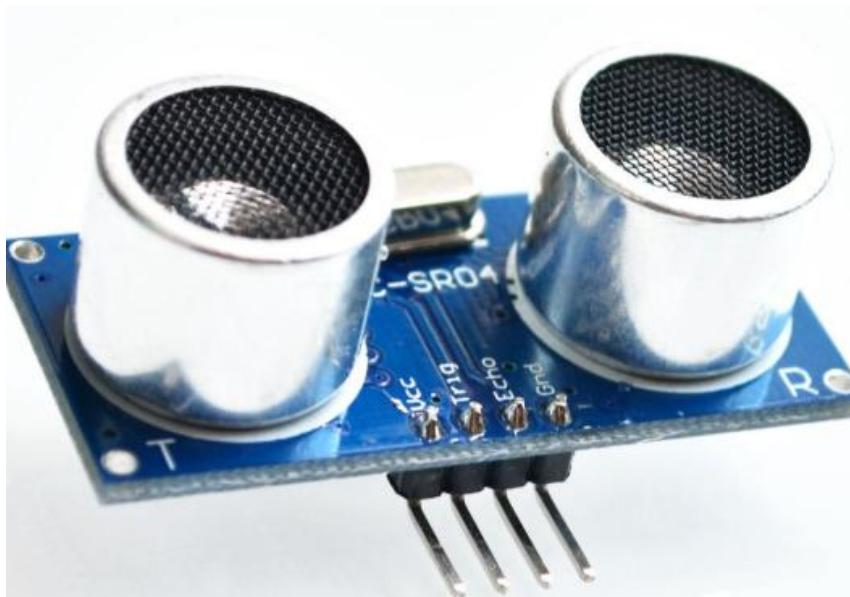


## **תדרונות מרכזיים של ESP32:**

- .**1. עוצמה חיובית**: עם מעבד כפול-ליבה ב מהירות של עד 240 MHz-ESP32-מאפשר ביצוע משימות מורכבות ועיבוד נתונים מהיר, מה שהופך אותו למתאים לפרויקט IoT מתקדמים.
- .**2. קישוריות אלחוטית מקיפה**: תמייהה מובנית ב- **Wi-Fi** ו- **Bluetooth** (כולל **BLE**) מאפשרת ייצור מגוון רחב של פרויקטים מוחברים, החל מממערכות בית חכם ועד לישומים תעשייתיים.
- .**3. גמישות בחיבורים**: מגוון רחב של ממשקים כמו **GPIO, ADC, DAC** ופרוטוקולי תקשורת כמו **I2C, UART, SPI** מאפשרים אינטגרציה קלה עם כמעט כל סוג של חישון או פעולה.
- .**4.יעילות אנרגטית**: מבצעי שינה מתקדמים וצריכת חשמל נמוכה מאפשרים ייצור מכשירים ניידים עם חיי סוללה ארוכים, מה שהופך אותו לאידיאלי למערכות ניטור מרוחקות.
- .**5. פיתוח נגיש**: תמייהה במגוון סביבות פיתוח כמו **ESP-IDF** ו- **Arduino IDE** וקהילת מפתחים גדולה ופעילה מקלים על תהליכי הפיתוח ומספקים שפע של ספריות תוכנה.
- .**6. יכולות IoT מתקדמות**: תמייהה בפרוטוקולים כמו **MQTT** ויכולות עדכון תוכנה מרוחק (OTA) מאפשרות הקמת שרתים **web** מקומיים וניהול התקנים מכל מקום בעולם.
- .**7. אבטחה משולבת**: תכונות אבטחה מובנות כמו הצפנה חזקה ו- **secure boot** חיוניות להגנה על מכשירי IoT ועל נתונים המשתמשים בעולם המחבר.
- .**8. חיישנים מיוחדים מובנים**: ה- ESP32 כולל חיישני מגע קפטייטיביים, המאפשרים ייצור ממשקים משתמש חדשניים ללא צורך ברכיבים נוספים.
- .**9. יכולות רשת מתקדמות**: תמייהה ביצירת רשתות **Mesh** ושימוש בפרוטוקול - **ESP-NOW** מאפשרות ייצור רשתות תקשורת מרובות ורב-כיווניות.
- .**10. התאמה למגוון יישומים**: מפרויקטים פשוטים ועד למערכות תעשייתיות מורכבות, ה- ESP32 מציע את הגמישות והעוצמה הדרושים למגוון רחב של יישומים.



## חישון מרחק – HC-SR04 :



ה-HC-SR04 זה החישון בו השתמשתי למדידת מרחק בפרויקט שלי. זה חישון מאד נפוץ בפרויקטים של רובוטיקה, חישוני חניה, ומכשורים חכמים. חישון זה נפוץ בגלל מספר דברים, כמו למשל: הוא נחשב לזול, פשוט לשימוש ומדויק יחסית בטעחים קצריים ובינוניים.

### איך החישון עובד?

העיקרון מאחורי ה-HC-SR04 מבוסס על גלי קול על-קוליים – חישון משדר גל קול בתדר גבוה, שאינו נשמע לאוזן האנושית (כ-40 קילו-הרץ). הגל הזה מתרחב בסביבה, וכאשר הוא פוגע במכשול או חפץ כלשהו, הוא חוזר אל החישון. החישון מודד את הזמן שלקח לגל להגיע חזרה. בעזרה מהירות הקול באוויר, ניתן לחשב את המרחק למכשול.

הנוסחה שמאפשרת לחשב את המרחק היא:

$$\text{מרחק} = \frac{\text{זמן החזרה} \times \text{מהירות הקול}}{2}$$

החלק החשוב הוא חלוקה ב-2, שכן הזמן שנמדד הוא הזמן הכלול של ההלוך והחזור של הגל. **מהירות הקול באוויר אינה קבועה**, והיא תלולה בעיקר בטמפרטורה. ככל שהטמפרטורה גבוהה יותר, המולקولات באוויר נעות מהר יותר ומעבירות את גל הקול ביעילות הרבה יותר. כתוצאה לכך, מהירות הקול עולה.

### נוסחה לחישוב מהירות הקול

ניתן לחשב את מהירות הקול באוויר (במטרים לשנייה) בצורה מוקorbit באמצעות הנוסחה הבאה :

$$\text{מהירות} = T \times 331.4 + 0.6$$

כאשר:

- $T$  היא הטמפרטורה במעלות צלזיאוס ( $^{\circ}\text{C}$ )
- $331.4 \text{ מ}/\text{s}$  היא מהירות הקול באוויר בטמפרטורה של  $0^{\circ}\text{C}$ .
- $0.6 \text{ מ}/\text{s}$  היא הקירוב לשינוי במהירות הקול עברו כל עלייה של מעלה אחת בטמפרטורה.

### השפעה על הדיווק

אם לא מתחשבים בטמפרטורה, חישוב המרחק יכול להיות לא מדויק.  
המערכות כה חשובות כגון כיפות ברזל או רובוט שומר בהתחשב בפרויקט שלי, יש צורך להתחשב בטמפרטורה הסביבתית על מנת להבין כיצד למרחק שמתקבל על פי החישון האולטרסוני.

### מבנה החישון:

- HC-SR04 מגיע עם ארבעה פינים עיקריים :
- VCC - מתח אספקה של 5 וולט.
  - GND - חיבור לאדמה.
  - Trig - פין טריגר, אליו שולחים פולס קצר (בדרכן כל 10 מיקרו-שניות) כדי להפעיל את שליחת הגל האולטרסוני.
  - Echo - פין החזרה, שופפיק פולס שמתממש לאורך זמן החזרה של הגל. אורך הפולס זהה מאפשר לחשב את המרחק.

### מגבלות ויתרונות

מצוין למדידה בטוחים של 2 ס"מ עד 400 ס"מ. עם זאת, יש לו מגבלות :  
משטחים קטנים או סופגים עלולים להקשوت על קריית המרחק, כמו גם רעים חזקים באוויר שמספריים לגלים האולטרסוניים. בנוסף, הוא רגיש לזווית החזרה של הגל – החישון עובד היטב כאשר המשטח ניצב אליו.

### מדידות מתח ואופן פעולה:

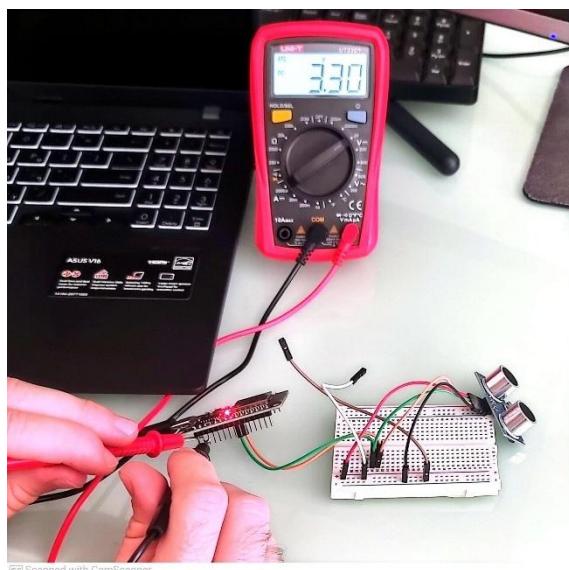
מדידת מתחים של החיבור האולטרסונייק אל המיקרו בקר ESP32, אראה את המדידה אל המתח אליו אני מחבר את החיבור עצמו אל המיקרו בקר. בנוסף לכך אראה את המתח שMOVEDה הפין של ה- 3.3V, האם הוא באמת מוצריה מתח זה:

### נראה תמונות:

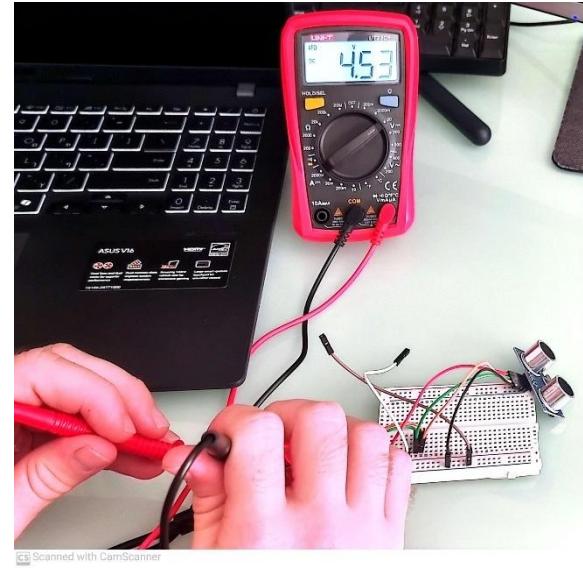
```

1 // חיבור תזוזאת בסיריאק HC-SR04-כ ESP32
2 // פולס-Trig פולס-Echo פין כ
3 #define TRIG_PIN 27
4 // כ Echo פולס-Trig פין כ
5 #define ECHO_PIN 26
6 // משנה לשינוי הזמן של גזען לפולס גזען
7 long duration;
8 // המרחק במטרים
9 int distance;
10
11 void setup() {
12     // פיתוחת תקשורת סיריאלית ליציאה בתוצאות
13     Serial.begin(115200);
14
15     // גדרת פינים
16     pinMode(TRIG_PIN, OUTPUT);
17     pinMode(ECHO_PIN, INPUT);
18
19     Serial.println(" מדידת מרחק עם חיישן אולטרסונייק ESP32");
20 }
21
22 void loop() {
23     // ל קיצור פולס-Trig
24     digitalWrite(TRIG_PIN, LOW);
25     delayMicroseconds(2);
26     digitalWrite(TRIG_PIN, HIGH);
27     delayMicroseconds(18);
28     digitalWrite(TRIG_PIN, LOW);
29
30     // כ-Echo HIGH כריאת מתח זמן ש
31     duration = pulseIn(ECHO_PIN, HIGH);
32
33     // חישוב מרחק בס"מ
34     distance = duration * 0.034 / 2;
35
36     // חישוב המרחק גמוך הסיריאלי
37     Serial.print("המץ: ");
38     Serial.print(distance);
39     Serial.println(" כ\");
40
41     // המתנה קצרה בין מדידות
42     delay(500);
43 }
```

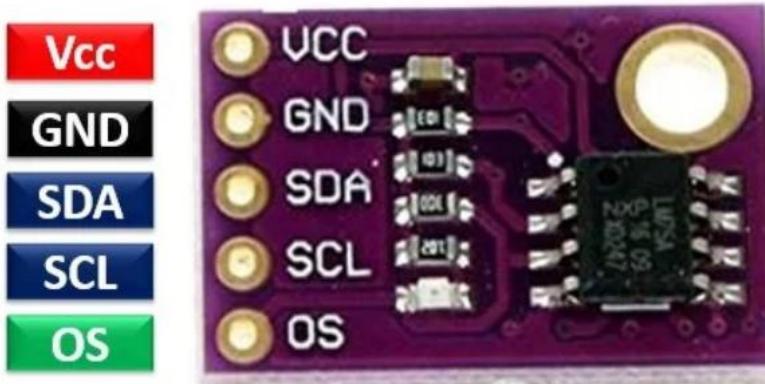
### המתח שMOVEDה המיקרו בקר מהפין 3.3V:



### המתח שאני מספק לחיפוי:



## חישון טמפרטורה LM75



חישון הטמפרטורה **LM75** אינו רק רכיב אלקטרוני פשוט אלא הוא מערכתי-על-שਬב (SoC) דיגיטלי קומפקטי שנועד למדוד טמפרטורה ולהמיר אותה לאוטות שמיקו-בקרים כמו הארדואינו או ה-ESP32 יכולים להבין בקלות. בנגוד לחישנים אנלוגיים (כמו ה-**LM35**) שמוסכאים מתח משתנה, ה-**LM75**- מבצע את ההמרה לאות דיגיטלי בתוכו, מה שפשט משמעותית את החיבור והתקנות.

### מאפיינים וдиוק טכני:

- **טוחה מדידה :** החישון מתוכנן למדוד טמפרטורות בטווח רחב של  $55^{\circ}\text{C} + \text{ עד } 125^{\circ}\text{C}$ . טוחה זה הופך אותו למתאים ליישומים תעשייתיים, רפואיים וביתיים.
- **רזולוציה ודיוק :** החישון מספק קריאה ברזולוציה של **9 ביט**, המקבילה לדיווק  $0.5^{\circ}\text{C}$ . דגמים מתקדמים יותר כמו ה-**LM75B** מציעים רזולוציה של **11 ביט**, מה שמספק דיוק גובה יותר של  $0.125^{\circ}\text{C}$  (למרות שהדיווק המוחלט עדיין תלוי בטווח הטמפרטורה).
- **צריכת אנרגיה :** יתרון משמעותי של ה-**LM75** הוא ייעילותו האנרגטית. הוא צורך זרם נמוך מאוד במצב פעולה, וככלל מצב **כיבוי** (Shutdown mode) בו הוא צורך זרם של מיקרו-אמפרים בלבד (בסביבות  $4\mu\text{A}$ ), מה שהופך אותו לאידיאלי ליישומים המופעלים על סוללות.
- **מתח עבודה :** החישון יכול לפעול בתחום שנו בין  $2.7\text{V}$  ל- $5.5\text{V}$ , מה שהופך אותו לתואם למגוון רחב של מיקרו-בקרים, כולל ארדואינו ( $5\text{V}$ ) ו-ESP32 ( $3.3\text{V}$ ).

### עקרון הפעולה והתקשות הדיגיטלית (I2C) : (כפי שהוסבר בתחילת הספר)

ה-**LM75** מתקשר עם המיקרו-בקר באמצעות פרוטוקול **I2C** (Inter-Integrated Circuit, במשמעותו TWI - Two-Wire Interface) גם בשם (I2C). פרוטוקול זה משתמש בשני קוויים בלבד לתקשורת:

- **SDA (Serial Data Line) :** קו דו-כיווני להעברת נתונים.
- **SCL (Serial Clock Line) :** קו שעון המסנכרן את התקשרות בין המיקרו-בקר לחישון.

לכל רכיב I<sub>2</sub>C יש כתובות ייחודית של 7 ביט. ה-LM75-מציע שלוש רגלי כתובות (A0, A1, A2) שנitin לחבר ל-VCC-או ל-GND, מה שמאפשר להגדיר עד שמונה **חישני LM75** שונים על אותו אוטובוס, I<sub>2</sub>C ללא הפרעות.

#### **כיצד זה עובד?**

המיicro-בקר (המאסטר) שולח בקשה לקריאת נתונים לכתובת הספציפית של ה-LM75-(העבד). החישן מגיב נתונים המאוחסנים בזיכרון הפנימי שלו, שם הטמפרטורה נשמרת באופן רצוף לאחר המרה.

#### **פונקציונליות "תרמוסטט-** (Over-temperature :

אחד התכונות המייחדות והשימושיות של ה-LM75 היא יכולתו לפעול כתרמוסטט דיגיטלי עצמאי. לחישן יש יציאה ייודית OS (Over-temperature Shutdown), שיכולה לשמש בשני מצבים :

- **מצב השוואה (Comparator Mode)**: המicro-בקר יכול להגדיר שני ספי טמפרטורה
  - בחישן עצמו :
  - TOS (Temperature Over-limit) : טמפרטורה מקסימלית. כאשר הטמפרטורה עולה מעלה מערך זה, יציאת OS-פעילה פلت דיגיטלי.
  - THYST (Hysteresis) : טמפרטורה שבה הפלט חוזר למצב רגיל. הדבר מונע הפעלה וכיבוי חזריים ונשנים (נדון) של היציאה כאשר הטמפרטורה נמצאת קרוב ל██.
- **מצב פסיקה (Interrupt Mode)** : במצב זה, יציאת OS- יכולה לייצר פסיקה (interrupt) למicro-בקר, להתריע בפניו על חריגת מהטמפרטורה, ללא צורך בסקרים מתמידים של החישן.

פונקציה זו שימושית במיוחד לבניית מערכות פשוטות של בקרת טמפרטורה, כמו מפוח שמופעל אוטומטית כאשר הטמפרטורה עולה על סף מסוים.

#### **שלבי חיבור בסיסיים:**

- .1 **VCC** (מתח) ו-**GND** (הארקה) : חיבור למקור מתח של 5V או 3.3V.
- .2 **SCL** ו-**SDA** : חיבור לפיני I<sub>2</sub>C המתאימים בארדואינו (ברוב הלוחות : A4 ו-A5 בהתאם) או ל-ESP32.
- .3  **נגד פול-אף** : יש צורך בנגדי פול-אף (בדרך כלל 4.7Ω) על קווי ה-SCL ו-SDA כדי למשוך את הקווים למתח גבוה כאשר הם אינם פעילים. במודולים רבים, הנגדים הללו כבר מובנים על הלוח.

#### **חיבור החומרה ב-ESP32**

חיבור הפיזי של חישן LM75 ל-ESP32 הוא פשוט מאוד, כיון ששניהם תואמים למתח עבודה של 3.3V.

- .1 **VCC**(מתח) של ה-LM75-מחבר לפין 3.3V ב-ESP32.
  - .2 **GND**(הארקה) של ה-LM75-מחבר לפין GND ב-ESP32.
  - .3 **SDA**(קו נתונים) של ה-LM75-מחבר לפין GPIO21 ב-ESP32.
  - .4 **SCL**(קו שעון) של ה-LM75-מחבר לפין GPIO22 ב-ESP32.
- פיני GPIO21 ו- GPIO22 הם בירית המחלל של I<sub>2</sub>C בלוחות ESP32 רבים, מה שמנשט את החיבור.

### שלבי תכונות (פסאודו קוד):

- .1. **הכללת ספרייה:** השתמש בספרייה ייעודית לחישון (למשל, Adafruit LM75) כדי לפשר את התקשרות.
- .2. **אתחול:** בפונקציית `setup()`, אתחול את החישון.
- .3. **קריאה נתוניים:** בפונקציית `loop()`, קרא את ערך הטמפרטורה מהחישון. הספרייה כבר תמיר את הערך הדיגיטלי לטמפרטורה בצלזוס.
- .4. **מצגת הנתוניים:** הדפס את הנתוניים לצג טורי (Serial Monitor) או לצג LCD.

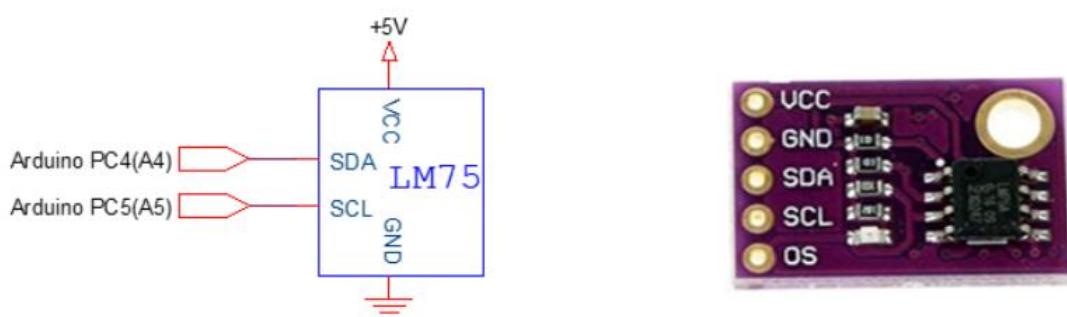
### ישומים נפוצים:

השילוב הייחודי של דיק סביר, צריכה נמוכה וממשק דיגיטלי פשוט הפך את ה-LM75 לבחירה פופולרית ב מגוון רחב של תחומיים :

- **מערכות קירור:** בקרת טמפרטורה בצד אלקטронני, שרתים ומחשבים כדי למנוע התחלומות יתר.
- **טרמוסטטים:** בניית מערכות בקרת אקלים פשוטות לבית או למשרד.
- **מכשירים ניידים:** שימוש במכשירים המופעלים על סוללות, כמו שעונים חכמים או מערכות ניטור אישיות.
- **מדידת טמפרטורה כללית:** ישומים מדעיים, תעשייתיים וחינוכיים שבהם נדרשת מדידת טמפרטורה אמינה ולא יקרה.

לxicom, ה-LM75 מספק פתרון אלגנטי וחסכוני למדידת טמפרטורה דיגיטלית. הוא משריר את המיקרו-בקר ממשימת המרת האות האנלוגי, מציע גמישות בשימוש בזוכות I<sub>2</sub>C ותכונות ה-"טרמוסטט" המובנות, מה שהופך אותו לרכיב יסוד בפרויקטים רבים של IoT ואלקטרונית.

### נראות ומאפיינים עיקריים של החישון:



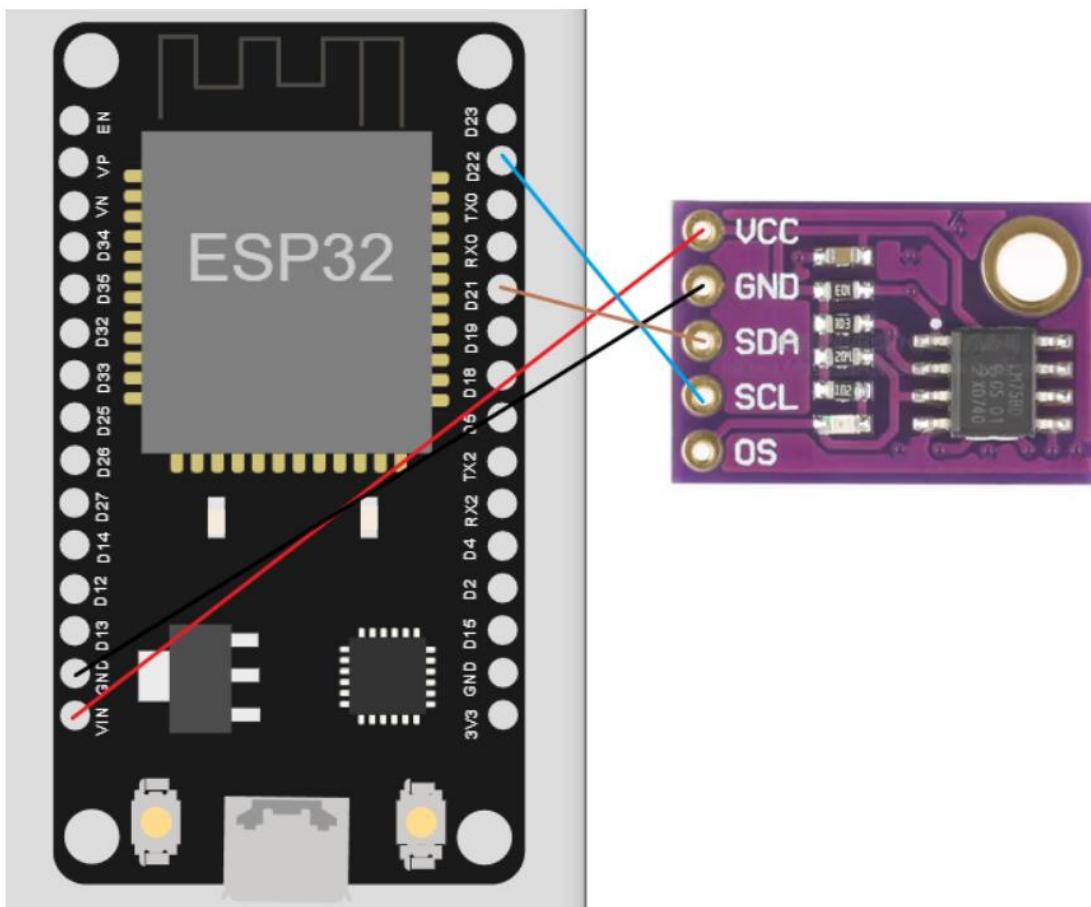
### הסבר מדוע החישון מחובר באמצעות ה프וטוקול I2C

I2C הוא פרוטוקול תקשורת שמאפשר לחבר התקנים שונים ללוח הארדואינו באופן פשוט יחסית. במקרה שלנו, החישון **LM75** מחובר דרך פיני ה- SCL ו- SDA של הלוח. פינים אלה משמשים לתקשורת I2C דרך הפינים האלה, הארדואינו ESP32 והחישון "מדוברים", באמצעות פרוטוקול I2C.

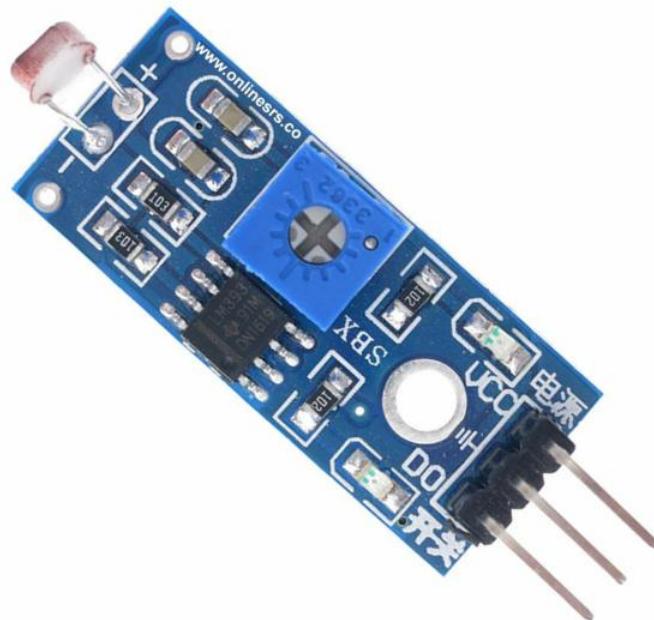
כשאנו רוצים לקרוא את הטמפרטורה, הארדואינו/ESP32 שולח בקשה לחישון דרך פיני ה- I2C. החישון קורא את הטמפרטורה, ושולח חזרה את הערך דרך אותם פינים. כך אנחנו יכולים לקרוא את הטמפרטורה בקלות רבה, ללא צורך בمعالג מורכב.

I2C מאפשר לנו לחבר מספר התקנים על אותם פינים SCL ו- SDA, ולתCKER איתם באופן עצמאי

### נראה חיבור של החישון LM75 אל מיקרו בקר ESP32:



## מודול חיישן האור KY-018



מודול חיישן האור (KY-018 או "LDR Module") הוא רכיב אידיאלי למדידת עצמת אור ובקרת תאורה אוטומטית באמצעות ה-ESP32. המודול משלב את החישון הבסיסי עם רכיבי בקרה, מה שמאפשר שתי דרכי עבודה: קריאה מדוקט או קריאה בינארית פשוטה.

### 1. המבנה והרכיבים העיקריים:

המודול בנוי סביב שלושה רכיבים מרכזיים:

1. **נגד תלוי-אור (LDR):** זהו החישן הראשי שאחראי על המדידה. כשהאור הפוגע בו גדל, החתנוגדות שלו יורדת. המודול מכיל מעגל פנימי שממיר את שינוי החתנוגדות הזה לשינוי מתח.
2. **קונפרטור מתח (LM393):** זהו הцип האלקטרוני הראשי. תפקידו הוא להשוות את המתח המשנה של ה-LDR מול מתח ייחוס קבוע (הסף). השוואה זו מניבה את הפלט הדיגיטלי (D0).
3. **פוטנציאומטר כחול:** זהו נגד משתנה שמאפשר למשתמש **לפייל ידנית** את מתח הייחס הקבוע של ה-LM393 ובכך לקבוע באופן חומרתי את סף ההדרקה/כיבוי של היציאה הדיגיטלית.

## **2. דרכי השימוש במודול (היציאות):**

הגמישות של המודול נובעת משתי יציאות הפלט שלו:

א. יציאה אנלוגית - (A) למדידה מדויקת

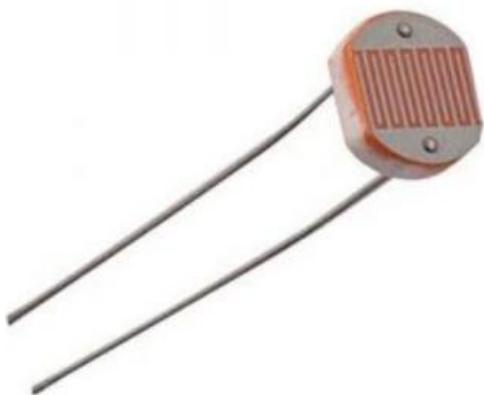
- תפקיד :** פין זה מספק את המתח המשתנה המגיע ישירות ממחלך המתח הפנימי של ה-LDR.
  - היתרון :** הוא מעביר את כל טווח האור באופן רציף (ערכיהם בטוווח של 0 עד 4095 ב-ESP32). שיטה זו מאפשרת לך לקבוע את סף ההדלקה **בתוך הקוד** בלבד, מה שנותן שליטה ודיוק גבוהים יותר.
  - חיבור :** חיבור את A0 לפין ADC של ה-ESP32 (למשל, GPIO34).

**ב. יציאה דיגיטלית - (D0) להחלטה בינהרית**

- **תפקיד:** פין זה הוא הפלט של ה-LM393. הוא מספק רק שתי אפשרויות: HIGH או LOW.
  - **היתרון:** השיטה היא "הכנס והפעל". ה-LM393 מבצע את החלטה האם חסוך או מואר עבורך, בהתאם לכיוול שקבעת באמצעות הפוטנצימטר הכהול.
  - **חיבור:** חיבור את D0 לפין **GPIO דיגיטלי** רגיל-ב-ESP32.

### **חישון אור - LDR - Light Dependant Resistor**

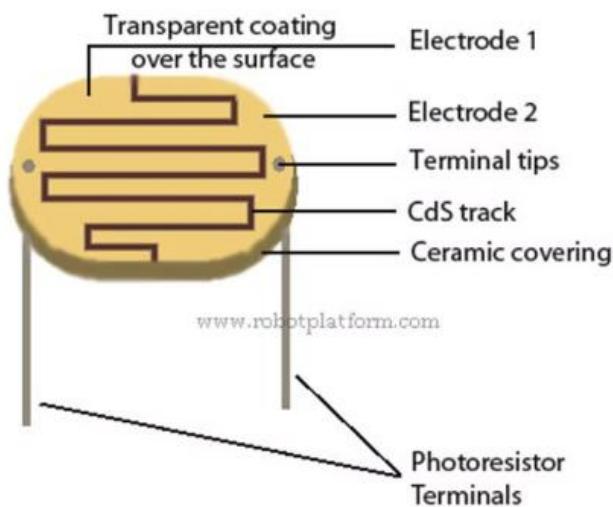
חישון היכול למדוד את עוצמת האור הנמצאת בסביבתו. שמו באנגלית מסגיר את אופן פועלתו - נגד תלוי אור, יכולומר נגד שמשנה את התנגדותו בתלות באור אליו הוא נחשף.



## מבנה חישון א/or:

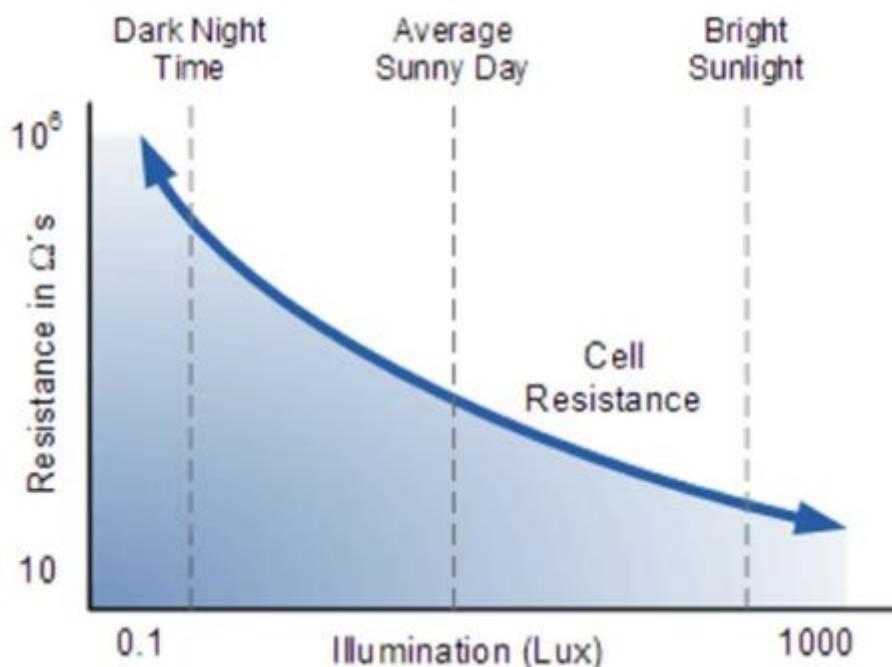
החיישן מורכב מודסquitת עגולה העשויה חומר קרמי. הדסקית מחולקת לשני אזורים המופרדים ביניהם על ידי חומר הנקרא קדמיום סולפייד CdS. חומר זה אחראי להתחנות החשמלית של החישון. משני החזאי הקרים יוצאים גם שתי קטבים מוליכים המתבחרים את המערכת.

- בחושך (התנודות גבואה):** במצב זה, האלקטרונים בחומר ה- CdS קשורים חזק לאטומים שלהם. כתוצאה מכך, ישנים מעט מאוד נושאי מטען חופשיים שימושיים לשעת זרם חשמלי, ולכן החישון מציג התנודות גבואה מאוד (עשרות קילו-אוום עד מגה-אוום).
  - באור (התנודות נמוכה):** כאשר אור (המורכב מפוטונים) פוגע בשטח הפנים של ה- CdS, הפוטונים מעבירים אנרגיה לאלקטרונים ומשחררים אותם מבננה האטום. שחרור זה יוצר מספר רב של אלקטרונים חופשיים.
  - תוצאה חשמלית:** ככל שנוצרים יותר אלקטرونים חופשיים, המוליכות של החומר גדלה, וההתנודות החשמליות של החישון יורדת באופן דרמטי.



#### אופין שינוי התנודות:

החישון משנה את התנודות בהתאם לעוצמת האור אליה הוא נחשף. כאשר החישון בחושך מוחלט איז התנודותו מאד גבוהה (ניתן לראות קטע מדף נתוני הייצור), ואילו כאשר הוא מואר התנודותו יורדת באופן מהיר. עוצמת תאורה נמדדת ביחידות מידת הנקראות Lux.



אוף המדידה של מודול חישון האור KY-018 מtabסס על עקרון **מחלק המתח**. הליבה של החישון היא **נגד תלוי אור (LDR)**, אשר התנודותו משתנה באוף הפוך לעוצמת האור. המודול מחובר ל-ESP32 דרך אספקת מתח של (VCC) 3.3V המודול עצמו כולל **נגד קבוע** בטור עם ה- LDR כדי ליצור מחלק מתח.

כחותאה משינוי התנודות של ה-LDR, המתח בנקודה המשותפת (המסומנת A0) משתנה. בקורס ESP32 מודד את המתח המשותפה זהה באמצעות הפין האנלוגי (A0) ומרתוגם אותו לערך מספרי בטוחה של **עד 4095**, בהתאם לרזולוציית 12bit של הממיר האנלוגי-לדיגיטלי (ADC) שלו.

## מסך TFT (ST7735) – 1.8



זהו המסך בו השתמשתי על מנת להציג את הרадאר בפרויקט שלי. מסך קטן וקומפקטי שיוצר תמונה ברורה וטובה עבור הראדאר. אשר מציג את המרחק החל מ 0 ועד 100 סנטימטרים. בנוסף ניתן לראות את הקו שמשתובב 180 מעלות החל מצידו הימני/הشمالي של המסך ועד לצידו השני ומציע נקודה בהתאם למרחק ככל שהאובייקט קרוב יותר נראה שצבעה של הנקודה הוא אדום, וככל שהאובייקט רחוק יותר הנקודה תהיה בצבע צהוב.

### **1. סוג המסך:**

- דגם : ST7735
- גודל : 1.8 אינץ'.
- רזולוציה :  $128 \times 160$  פיקסלים.
- צבעים : צבעוני (RGB565).
- ממשק : SPI (Serial Peripheral Interface) מהיר ומדויק.

### **2. חיבור פיניים (SPI):**

המסך מתקשר עם ה- ESP32 דרך SPI. בפועל בקורס שלי :

- MOSI (Master Out Slave In) – GPIO 23 (נתונים מה- ESP32 למסך).
- SCLK (Clock) – GPIO 18 (שעון SPI).
- CS (Chip Select) – GPIO 5 (בחירה התקן SPI פעיל).
- DC (Data/Command) – GPIO 2 (מספר למסר “פוקודה או נתון”).
- RESET – GPIO 4 (Ấתחול המסך).
- VCC / GND - מתח והארקה.

**הערה:** SCLK ו-MOSI – משותפים אם יש יותר מסך אחד CS ו- DC - חיבורים להיות שונים לכל מסך.

### 3. תאורת רקע (Backlight)

- לרוב המטכים יש **LED backlight** פנימי.
- חיבור פיזי: - BL / LED+ / LED.
- אם מחברים ל- 5V → Backlight.
- ניתן לכבות או לשנות בעוצמה דרכן טרנזיסטור PWM / כדי לחסוך צריכה חשמלית.
- כיבוי Backlight חיסכון משמעותי בזרם. (50–300 mA).

### 4. ספריות ותפקודים בקוד שלבי:

- הספרייה Ucglib שולחת פקודות SPI למסך.
- פונקציות עיקריות בקוד:
  - ucg.begin() - אתחול המסך.
  - setRotate90() - סיבוב המסך לאופקי.
  - setColor(...) - הגדרת צבע לצירוף.
  - drawLine, drawBox, drawDisc, drawCircle - ציור צורות.
  - setFont(...), setPrintPos(...), print(...) - כתיבת טקסט.
  - cls() - ניקוי המסך באמצעות ריבועים שחורים.

### 5. זרם ומתח

- VCC : רוב המודולים מקבלים 5V חלק תומכים גם ב-3.3V.
- צורך טיפוסית:
  - backlight: 10–60 mA
  - backlight: 50–300 mA

### מה המסך מציג בקוד שלבי:

- רקע גрафי**: עיגולים, קוויים, רקע ירוק-שחור (גרדיינט)
  - טקסט**: שם הפרויקט "Mini Radar", מרחקים בקנה מידה (...) (25cm, 50cm...).
  - נקודות**: מיקום אובייקטים לפי החישון האולטראשוני (>100cm <100cm, אדום <צהוב).
  - קוויים**: קו סריקה שמסתובב עם הסרוון, כמו קרן רdar.
- בקיצור, המסך הוא **המסך הגרפי של הרdar** – מציג מידע בזמן אמת בצורה צבעונית.

### הסביר על תוצאות הרadar:

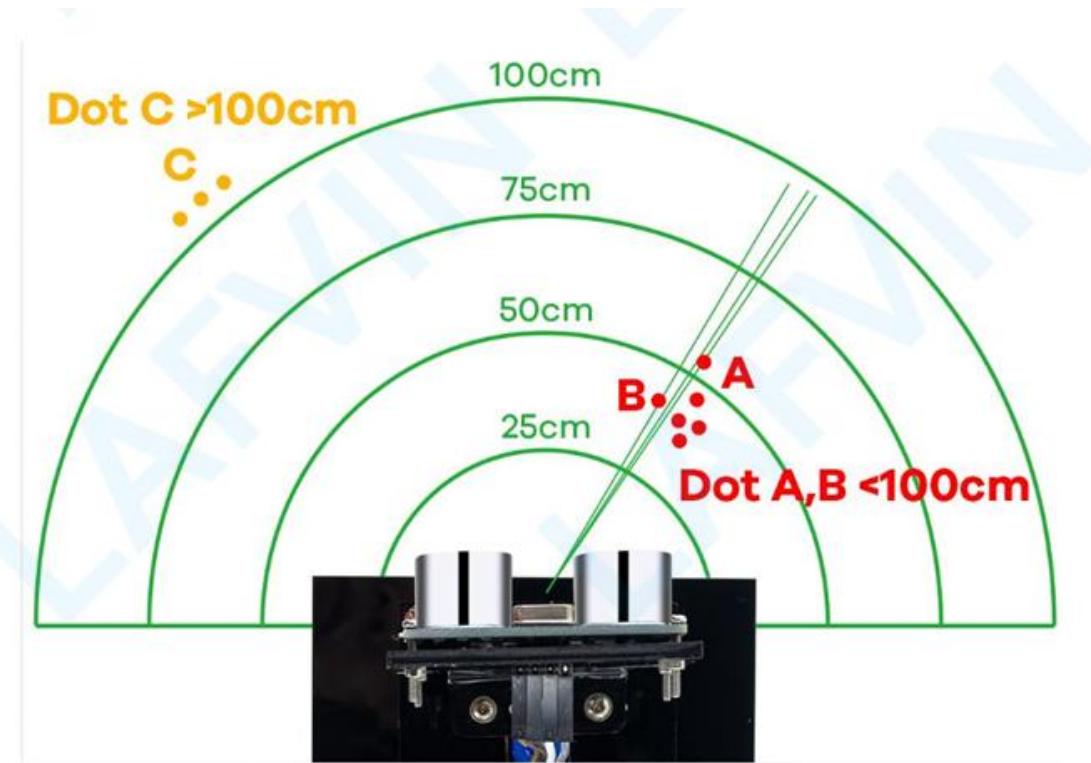
#### **איך למשה המרצת עובדת מה אנחנו רואים על המסך?**

אוначיל בזזה שהרוביוט השומר שלנו צריך לשמור על סביבה שנסרקת 180 מעלות.

הפעולה של הסריקה עצמה מתבצעת באמצעות שני רכיבים חשובים מאוד במערכת radar: הרכיב הראשוני הוא המנוע סרוי שבכלל מאפשר לנו את כל העניין של התזוז של חצי הסיבוב החל מ0 מעלות ועד הגעתו ל180 מעלות. הרכיב השני שלנו שלמעשה מבצע את המידה זה בחישון האולטראוניק שמבצע בדיקה של האובייקטים בסביבה בתווך של 100 פלוס סנטימטרים ומחזיר לנו "מידע" שלמעשה מתאפשר בנקודה על radar.

כל המערכת מוצגת באופן חד וברור על המסך שלו ואנו יכולים לדעת האם האובייקט קרוב אלינו או רחוק מיתנו, כאשר האובייקט נמצא בתווך בין 0 ל- 100 סנטימטרים נראה שהנקודה היא קבוע אדום. כאשר האובייקט נמצא בתווך של 100 סנטימטרים או הנקודה תהיה קבוע צהוב.

**נראה תמונה שמחישה לנו באופן ברור את מה שתרחש במסך בזמן פעולה מערכת radar:**



כמו שאמרתי קודם לכן, ניתן לראות מההמונה שכל עוד אנו נמצאים בתחום בין 0-100 סנטימטרים固定 הנקודות האו אדום וברגע שאנו עברים תחום זה固定 הנקודות הוא צהוב.

## מסך גרפי טאץ' TFT 9341

מסך גרפי 9341 TFT מאפשר הצגת תווים, מספרים, צורות גרפיות ותמונות על המסך.

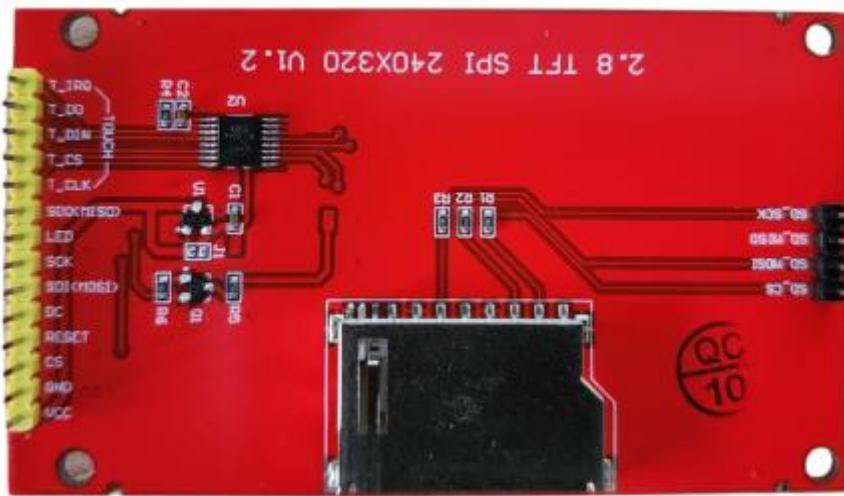


### מפורט טכני של מסך גרפי:

- .**1. סוג תצוגה :** מסך גרפי צבוני מגע (Touch).
- .**2. פרוטוקול עבודה :** SPI (Serial Peripheral Interface).
- .**3. גודל פיזי 8 :** אינץ'.
- .**4. רזולוציה 240x320 :** פיקסלים (נקודות).
- .**5. ציר X רוחב :** (מכיל 320 פיקסלים).
- .**6. ציר Y גובה :** (מכיל 240 פיקסלים).
- .**7. מתח הפעלה :** 3V.
- .**8. תכונת אחסון :** תצוגה כוללת קורא כרטיסי SD המאפשר שמירת קבצי תמונה בנפח זיכרון גדול.

**הערה :** אם תעדיף שהסדר יהיה רוחב ואז גובה (כפי שמקובל לעתים ברזולוציה), ניתן לשנות את סעיפים 5-6 בהתאם לצורה שבה תרצה להציג את זה במסך. הרשימה הנוכחית משקפת את המידע שסבירת, כאשר ציר X הוא 320 וציר Y הוא 240.

## תיאור ותפקיד פיני חיבור – מסך TFT/Touch :



המסך הגרפי דורש מספר חיבורים מרכזיים כדי לאפשר תקשורת נתונים מהירה (SPI) ובקרה מגע. כל פין משרת מטרת ייחודית:

- 1. VCC מתח אספקה :** זהו פין אספקת המתח החיובי של התצוגה, הנדרש להפעלת הבקר והתאורה האחוריית. עבור מסך זה, **המתח הנדרש הוא 3.3V**.
- 2. GND הארקה :** (חיבור הארקה המשותף, סוגר את המעגל החשמלי).
- 3. CS בחירת רכיב - (Chip Select) :** משמשת לשיליטה על שבב התצוגה (LCD). כאשר הפין זהה במצב נמוך (LOW) לוח הבקרה (כمو ESP32) מסנו לבקר התצוגה (ILI9341) שהוא הרכיב העיקרי לקבל או לשלוח נתונים דרך דרכ קו-ה-SPI המשותף.
- 4. DC נתונים/פקודה (Data/Command) :** – פין קריטי בבקרה. הלוח משתמש בו כדי להבדיל בין סוג המידע הנשלח: אם הפין במצב גבוה (HIGH) המידע הנשלח הוא **נתונים** (צבע פיקסלם, תמונה); אם הפין במצב נמוך (LOW) המידע הוא **פקודה** (הוראות לבקר לשנות רזולוציה או אתחול).
- 5. MOSI - Master Out Slave In (SDI) :** קו **כניסת הנתונים הראשי** בפרוטוקול SPI. באמצעות פין זה, לוח הבקרה שולח את כל הנתונים הגרפיים והפקודות אל התצוגה.
- 6. CLK שעון - (Clock) :** רgel השעון. מספקת את הקצב והתיזמון הנדרשים לסינכרון תהליך העברת הנתונים ב-SPI. כל פיקסל או בית נשלח בתיאום עם דופק השעון.
- 7. MISO - Master In Slave Out (SDO) :** קו **יציאת הנתונים** מפרוטוקול SPI. דרכ פין זה, לוח הבקרה **מקבל נתונים** חזירים מהמסך (למשל, קריית סטטוס או זיהוי הבקר).
- 8. CS בחירת שבב מגע :** (זהו פין Chip Select נפרד המשמש לשיבב בקר המגע לרוב (XPT2046) הואאפשר ללוח הבקרה לתקשר עם שבב המגע באמצעות רשת SPI, בנפרד מבקר התצוגה הראשי.
- 9. T-IRQ\_F- פסיקת מגע Touch Interrupt :** רgel זו נכנסת לפעולה כאשר **נגיעה מזויה** על פני המסך. היא מאפשרת ללוח לעבד אירועי מגע באופן יעיל ומידי (mbased פסיקה), מבלי צורך לבדוק באופן קבוע אם התרחש נגעה.

## הסבר מקייף על פונקציות התצוגה (Adafruit ILI9341): בה אני משתמש בפרויקט

### 1. הכללת הספריות (Headers):

בתחילת כל סקיצה (קוד) עליך לכלול את הספריות הבאות, המגדירות את כל הפקודות הגרפיות והחומרתיות:

- `#include <Adafruit_GFX.h>` : זהה ליבת הציור הגרפי (GFX Core). היא מספקת את כל הפקודות הגרפיות לצור צורות, קוים וטקסט, ואני תליה בברcker המסך הגרפי.
- `#include <Adafruit_ILI9341.h>` : זהה הדרייבר הגרפי התומך בברcker הפיזי ILI9341 של המסך שלו. הוא מתרגם את הפקודות הגרפיות של GFX לפקודות בرمאה נמוכה (SPI) שהחומרה מבינה.

### 2. אתחול ותחילת עבודה:

- פונקציות אלו נראות לרוב רק פעם אחת בתחום פונקציית `setup()`:
- `() tft.begin()` : זהה הפקודה הראשונה והקריטית ביותר. היא שולחת את כל רצף האתחול הנדרש לבקר ILI9341 כדי להביא אותו במצב עבודה וכיוול ראשי. (מקביל ל- `lcd.begin()`)
  - `() tft.setRotation(0)` : כיוון (פקודה זו קובעת את **כיוון התצוגה** (אורינטציה). ישנו ארבעה מצבים אפשריים (0, 1, או 2, או 3), כאשר אחד מהם יגדיר את המסך לרוחב (320x240) והאחר לאורץ (240x320).
  - `() tft.fillScreen()` : צבע (זויה הפקודה **לנקיי המסך**. היא ממלאת את כל שטח התצוגה בצבע אחד, ומשמשת לרוב לקביעת צבע הרקע (למשל, lcd.clear() מחליף `tft.fillScreen(ILI9341_BLACK)` במערכות אחרות).

### 3. עבודה עם טקסט ומיקום:

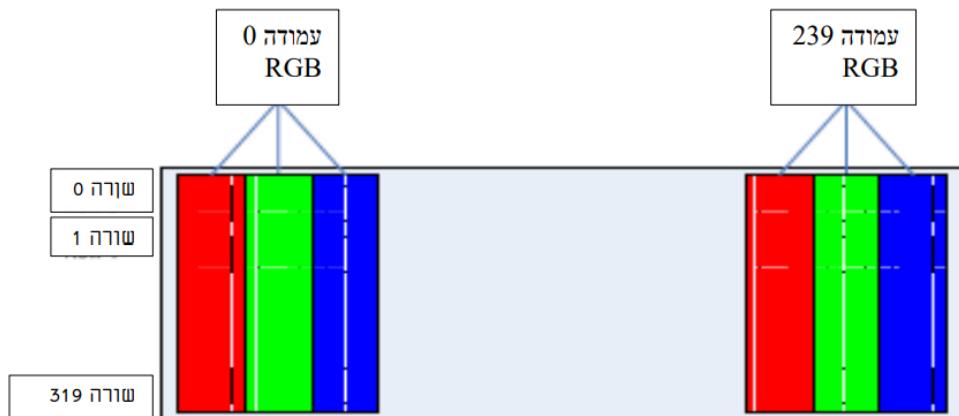
- פונקציות אלו משמשות להציג נתונים מסוימים ומספריים ומילוליים על המסך:
- `(y) tft.setCursor(x, y)` : מגדירה את מיקום הסמן שבו הטקסט הבא יודפס. ה- X וה- Y מציננים את הנקודה השמאלית העליונה של האות הראשונה. (מקביל ל- `lcd.goto(x, y)`)
  - `() tft.setTextSize(g)` : קובעת את גודל ה폰ט. גודל 1 הוא הפוןט הבסיסי, וגדלים גדולים יותר מכפילים את הפיקסלים (לדוגמה, גודל 3 הוא פי 3 מהגודל הבסיסי).
  - `() tft.setTextColor(c)` : צבע (קובעת את **צבע הטקסט** (צבע החזית) שיוצג. ניתן להוסיף ארגומנט שני אופציוניaldi כדי להגדיר את צבע הרקע סביב הטקסט.
  - `() tft.print(s)` : מבצעת את **הדפסת הטקסט, המספרים או המשתנים** אל המסך במקום הסמן הנוכחי (mostr מוסיפה ירידת שורה). (מקביל ל- `lcd.print(s)`)

### 4. פונקציות מגע (Touch):

- הפונקציות הללו מטפלות בברcker המגע (ככל הנראה XPT2046 הנמצא במכשיר):
- `() ts.begin()` : אתחול שבב המגע (כאשר ts הוא אובייקט של ספריית המגע הנפרדת, כמו XPT2046\_TouchScreen). זהו המקביל לאתחול בקרת המגע.
  - `() ts.getTouch()` : פונקציות ספציפיות בספרייה המגע, (כגון: ts.getTouch()) מאפשרות **lezehot at negiha** ולתקבל את קואורדינטות ה- X וה- Y של נקודת הלחיצה על המסך.

### תצוגות - TFT טרנזיסטור סרט זק – הן תצוגות אקטיביות:

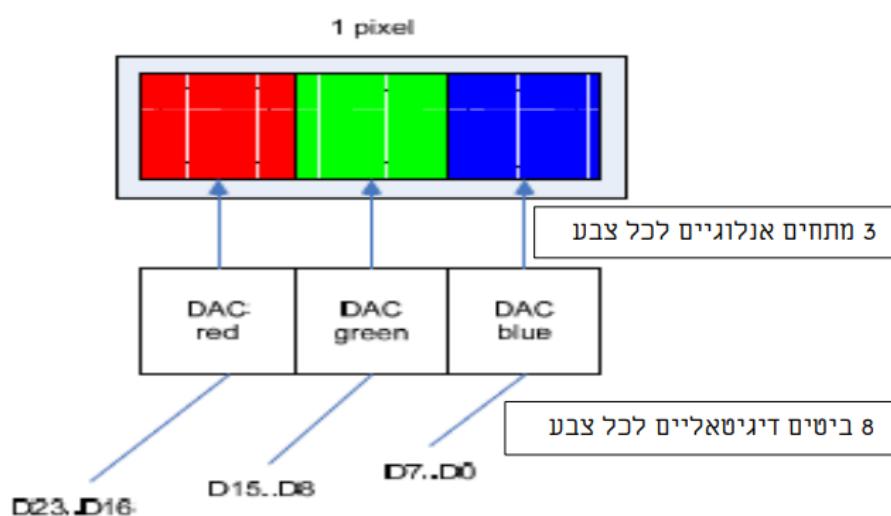
כל סגמנט בתצוגה מאפשר החזרה או העברה של אור. גם כאן יש מקור אור המקרין אוור מאחרוי התצוגה אל החזית. בעזרת הטרנזיסטור קבועים האור יוקרן החזית או שהוא ייחסם ולא יועבר. כדי ליצור צבעים צריך 3 סגמנטים. 3 סגמנטים אלו מעבירים אוור דרך מסנתת אדומה, יroxה או כחולה. כך נוצר פיקסל RGB. כלומר פיקסל מורכב מ- 3 סגמנטים. למעשה כל פיקסל מורכב מ- 2 מסננות עם קיטוב, 3 מסננות צבע ו 2 שכבות יישור וכך נקבע בדיקוק כמה אוור עברו ובאיזה צבע. באирור רואים את סידור העמודות והשורות של תצוגת RGB של 320 שורות ו- 240 עמודות.



### צירת צבע בתצוגה TFT (Thin Film Transistor)

تצוגות TFT יכולות לדוחוף 3 סגמנטים (פיקסל אחד) בpolloס שעון אחד עם שדה חשמלי משתנה חזק. גודל המתח האנלוגי בכל צבע קובע את עצמת הצלב המסוים. תצוגה כזו תומכת בצבעים רבים. רמות הצלב נקבעות על ידי מספר קווי הנטוון בפנל התצוגה ובמספר קווי יציאות הנטוון של הבקר. האפשרויות הן: 24 בית לפיקסל (bits per pixel), 15bpp, 16bpp, 18bpp, 8bpp. במשך מקבילי נדרשים 320 פולסי שעון ל- 320 פיקסלים.

ניתן לראות בתמונה מושך של 24 ביטים לפיקסל בשעון אחד:

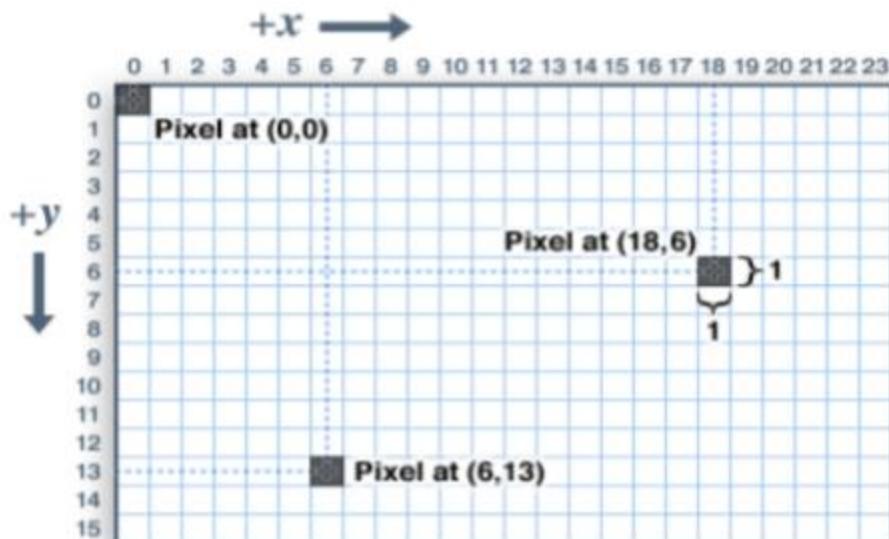


תמונה זו מותארת תצוגה המתחברת בMUX של 24 ביט.

כל 8 ביטים מתחברים ל DAC הקובע את גודל המתח שיגיע לכל צבע בפיקסל מסוים. הצלע הכחול נשלט על ידי 8 הביטים , D0~D7 הצלע הירוק על ידי הביטים D8~D15 והצלע האדום על ידי D16~D23. 8 הביטים של המספר הדיגיטלי קובעים את גודל המתח האנלוגי שיוצא מהוואיתו את עצמת הצלע בכל אחד מ 3 צבעי ה-RGB.

התמונה בתצוגת TFT מורכבת מפיקסלים. תצוגת TFT זו מורכבת ממכפלת של 2 המספרים 320 × 240 (הנותנת 76800 פיקסלים). ניתן לשולט על הצלע של כל פיקסל ומכאן ניתן להבין כי ניתן להציג תצוגה מנוקודה בודדת ועד תמונה מורכבת.

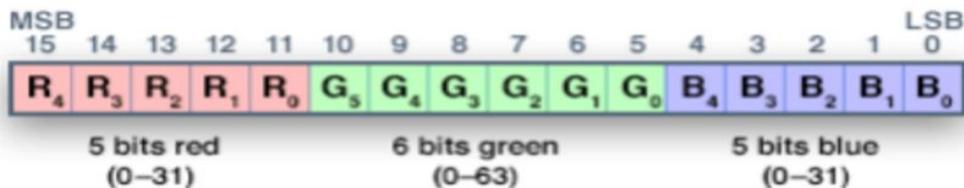
לכל פיקסל יש קואורדינטות (קואורדינטות הן קבוצת מספרים המציינות את מיקומו של גוף במרחב כלשהו) של x ושל y שלו. מערכת הקואורדינטות ממקמת את הראשית (0,0) (בפינה השמאלית העליונה כאשר x גדול לכיוון ימין ו- y כלפי מטה, ניתן לראות באירוע).



בתוצאות התומכות בצבע, הצבע נרשם כערך בן 16 סיביות לא מסומן – (unsigned) כמו כן רק חיוبي). בחלק מהתוצאות ניתן לקבל יותר צבעים ובחילוק פחות – יותר או פחות סיביות בערך. רוב הספריות שעובדות עם תוצאות כאלה עובdot עם 16 סיביות. גם לארדואינו קל לעובdot עם צבעים בני 16 ביטים. שלושת צבעי היסוד -RGB- אדום (red), ירוק (green), כחול (blue) נוכנים בתוך ה-16 סיביות בצורה הבא:

#### מהאיור רואים:

5 הביטים הגבוהים הם של הצבע האדום והם בני 5 ביטים (0-31).



6 הביטים הבאים הם של הצבע הירוק (0-63).

5 הביטים הנמוכים הם של הצבע הכחול (0-31).

צבע הירוק יש 6 ביטים כי העין של האדם רגישה יותר לצבע הירוק,

לדוגמה, אם נרצה לצבע פיקסל בצבע ירוק נשלח את הערך  $00000\ 11111\ 00000 = 07E0H$  הערך:  $F800H = 1111100000000000$  יציב את הפיקסל באדום והערך FFFFH יציב את הפיקסל לבן.

#### הצבעים הנפוצים הם:

- #define BLACK 0x0000 • שחור //
- #define BLUE 0x001F • כחול //
- #define RED 0XF800 • אדום //
- #define GREEN 0x07E0 • ירוק //
- #define CYAN 0x07FF • טורקיז //
- #define MAGENTA 0XF81F • גוון אדום //
- #define YELLOW 0xFFE0 • צהוב //
- #define WHITE 0xFF • לבן //

```

1. // Color definitions
2. #define BLACK      0x0000      // שחור
3. #define BLUE       0x001F      // כחול
4. #define RED        0XF800     // אדום
5. #define GREEN      0x07E0     // ירוק
6. #define CYAN       0x07FF     // כחול ירוק
7. #define MAGENTA    0XF81F     // גוון אדום
8. #define YELLOW     0xFFE0     // צהוב
9. #define WHITE      0xFF      // לבן

```

בPRIOSHOM הקוואורדינטאות , קודם נרשמת הנקודה בציר ה X של הפיקסל ולאחריה בציר ה Y .  
הקוואורדינטאות מבוטאות ביחידות של פיקסל ולא בגודלים של אורך (מ"מ או ס"מ אינץ'ים וכו').  
גודל מלבן שנצייר בהמשך יהיה במספר פיקסלים שלו בציר ה X ומספר הפיקסלים בציר Y ולא  
באורך של מ"מ או ס"מ. ניתן למדוד את אורך ורוחב המשך ואז לחלק בכמות הפיקסלים בכל ציר  
ולדעת את הגודל של כל פיקסל, אם כי הדבר פחות שימושי. ניתן לשנות את האוריינטציה של  
התמונה כך שהאורך יהיה רוחב והרוחב יהיה אורך.



נקבל תמונה כזו :



כלומר ממצב של תמונה כזו :

התצוגה הימנית נראית פורמט landscape (נוף) והשמאלית portrait (דיוקן). ראשית  
הצירים היא הנקודה השמאלית למעלה. ניתן לקבל 4 סוגים אוריינטציה, ככלומר איזו אחת מ 4  
הפינות של המלבן תהיה ראשית הצירים (0,0) .

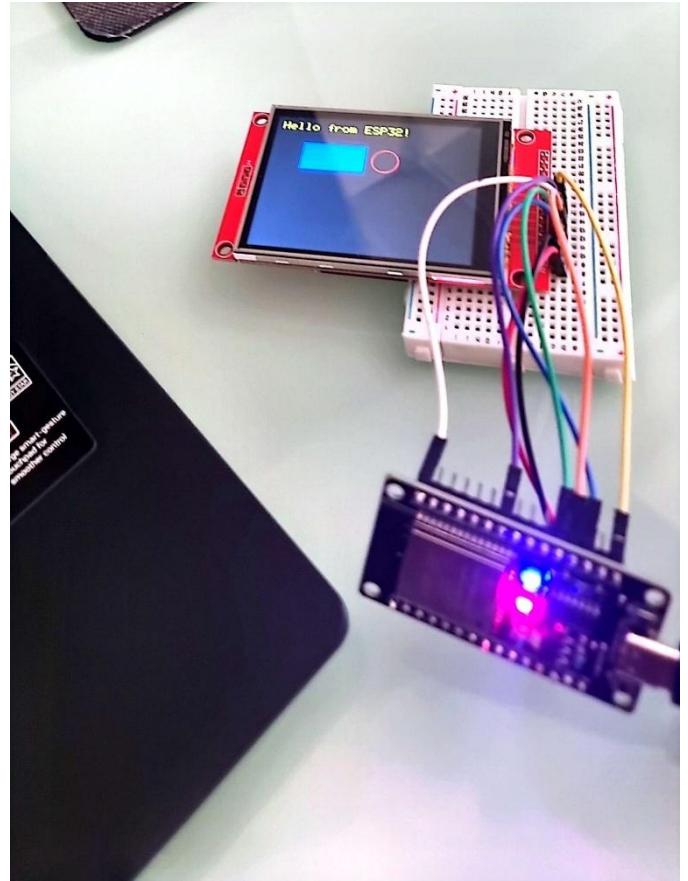
פונקציות הגרפיקה לכל תצוגת TFT יש בקר תצוגה פנימי המנהל את התצוגה עצמה ואת הקשר  
עם העולם שמחוץ לתצוגה שבו יש בדרך כלל מיקרו בקרים השולחים פקודות להפעלת התצוגה.  
כל בקר יש פונקציית אתחול משלה. כדאי לראות בדף הנtones של התצוגה מי הבקר המפעיל  
אותה. במידה ולא יודעים אז בפונקציית.

#### נראה בעט אופן פעולות המשך ומדידת המתחים בפנים אליהם אני לחבר את המשך :

```

1 #define TFT_CS 15 // Chip Select (CS)
2 #define TFT_DC 2 // Data/Command (DC)
3 #define TFT_RST 4 // Reset (RST)
4 // הכללת הספריות
5 #include <Adafruit_GFX.h>
6 #include <Adafruit_ILI9341.h>
7 // יצירת אובייקט המשך
8 // אנו מתייחסים לו את הגדרות הינוויים הנוטפים
9 Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC, TFT_RST);

10 void setup() {
11   Serial.begin(115200);
12   // אתחול המשך
13   tft.begin();
14
15   // (3 ו 2, 1, 0) הגדרת כיוון המשך
16   tft.setRotation(1);
17
18   // ייקוי המך ומילוי בצבע שחזור
19   tft.fillRect(0, 0, 160, 160, ILI9341_BLACK);
20
21   // הגדרת גודל וצבע הטקסט
22   tft.setTextSize(2); // 2 : גודר טקסט
23   tft.setTextColor(ILI9341_YELLOW); // צבע טקסט: צהוב
24
25   // (x, y) הגדרת טקסט בנקודה
26   tft.setCursor(10, 10);
27   tft.print("Hello from ESP32!");
28
29   // רוזב, גובה, צבע (x, y, a) צייר מלבן חלא
30   tft.fillRect(50, 50, 100, 50, ILI9341_BLUE);
31
32   // רוזב, גובה, צבע (x, y, a) צייר מעגל
33   tft.drawCircle(180, 75, 20, ILI9341_RED);
34
35 }
36 void loop() {
37 }
```



Scanned with CamScanner

מתמונה זו ניתן לראות שהדפסתי על המסך דברים בסיסים על מנת לראות שהוא פועל יחד עם המיקרו בקר ESP32 מה שניתן לראות למעשה על המסך מה שמודפס עליו זה Hello from ESP32 בנוסף מלון ממולא בצבע כחול ומעגל שرك החיקף שלו צבוע בצבע אדום ואינו צבוע מבפנים.

הקוד הוא פשוט ובסיסי רק כדי להמחיש את אופן הפעולה של המסך יחד עם המיקרו בקר ולראות שהוא מדפיס ואכן מתבצעת תקשורת בין המסך למיקרו בקר.

כעת נראה את מדידת המתחים:

מדידת מתח הנכנס לפין LED:



Scanned with CamScanner



Scanned with CamScanner

בהמשך הספר תחת הכותרת "תיעודים" ניתן לראות הסבר מדוע המתח אינו 5 וולט אלא בקירוב 4.5 וולט. בחישון אולטרסונייק כאשר ביצעתי מדידה של המתחים מהפינים האלו במיקרו בקר אליו נכנסים הפינים VCC ו-GND ברגע שמדדתי בפין ה-VN. קיברתי גם מתח דומה למתח זה ובפין ה-3.3V קיברתי גם בקירוב למתח 3.3V. חקרתי מדוע תופעה זאת קורת והסבירתי בתיעוד הראשון תחת הכותרת "תיעודים" בתיאור 5.9.25, הסברתי מדוע תופעה זאת עלולה לקרות ומה הסיבות שגורמות ל"איבוד" המתח בכניסה VN.

# מנוע SERVO

## הסביר כללי על המנוע SERVO:

מנוע SERVO הוא מנוע שבו יש את המרכיבים הבאים :

- **מנוע זרם ישיר (DC Motor)**
- **מערכת תמסורת פנימית של גלגלי שיניים**
- **בקשה אלקטרוניות על מיקום המנוע.**

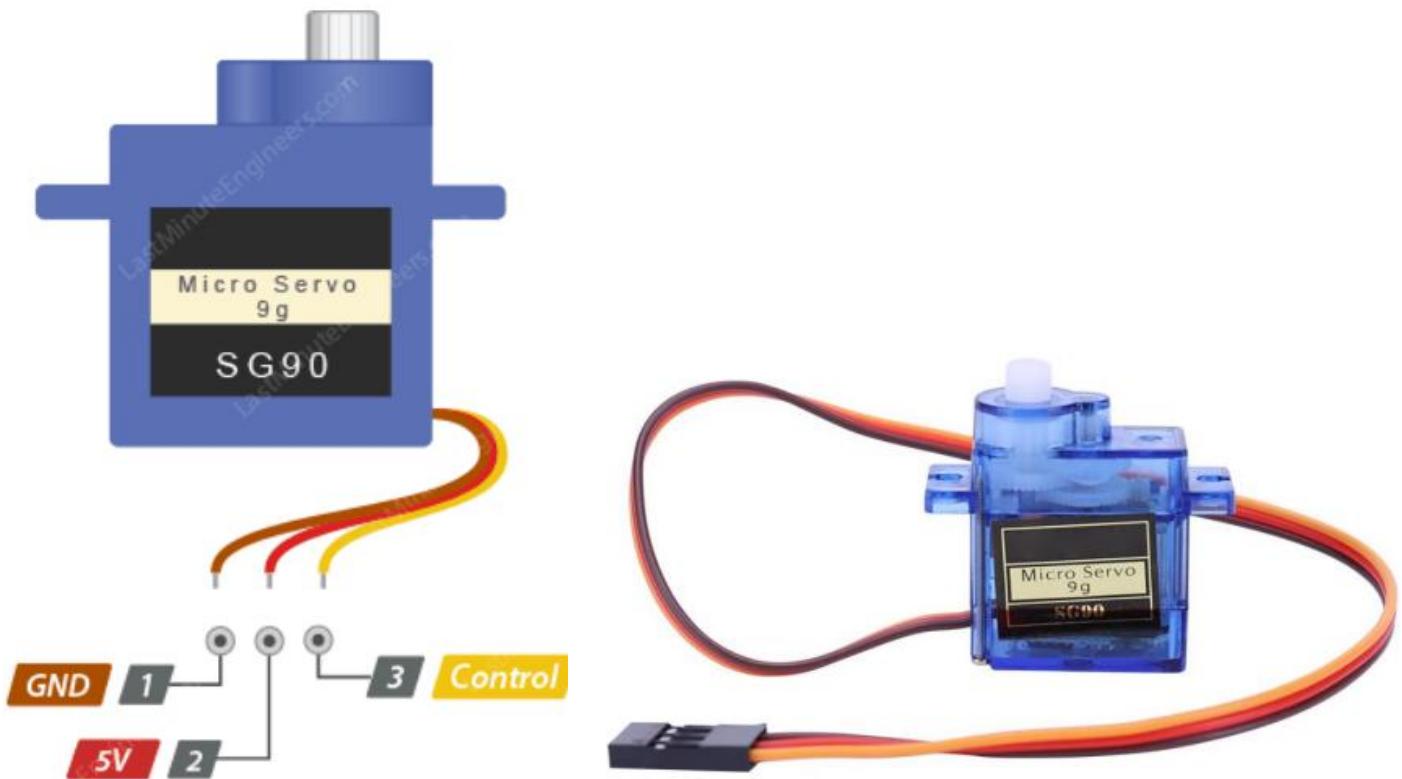
מנוע SERVO לא מסתובב בצורה חופשית כמו מנועי DC אלא נע עם פי זווית – לרוב בין 0 ל- 180 מעלות (אם כי יש מנועי SERVO שמסתובבים גם עד 360 מעלות).

מנועי SERVO נפוצים מאוד ומשמשים באפליקציות רבות. הסיבות לכך הן : יכולת השיליטה על **מנועי SERVO**, דרישות האנרגיה הנמוכות) **יעילות**, (ה**כוח החזק יחסית** של המנוע, **רמת המתוח** המשמשים להפעלו, **הגודל, המשקל והמחיר** הנמוכים שלו.

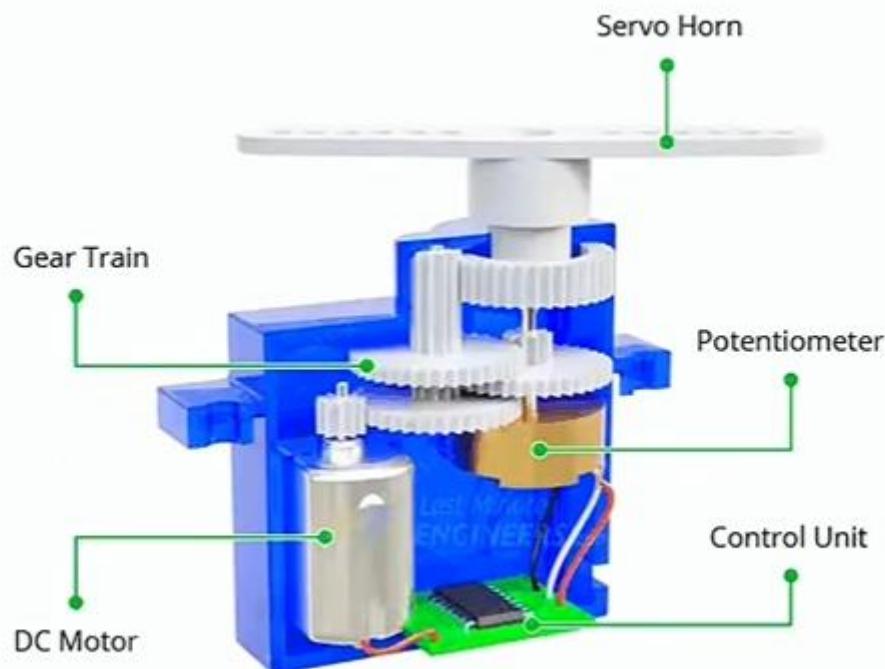
מנועי SERVO פועלם **בתוך מעגל סגור**, כלומר יש להם מערכת בקרה על מיקום המנוע והם יכולים לתקן הפרשנים מהמיוקם הרצוי.

האיור הבא מתאר מנוע SERVO שנitinן להשיג באינטרנט במחירים נמוכים יחסית ואת החיבורים שלו המנוע נקרא : **SG90**.

## נראה כיצד נראה המנוע SERVO:



מנוע SERVO מכיל מנוע DC קטן המחבר לציר היציאה דרך גלגלי השיניים. על הציר מחובר פוטנציאומטר המחזיר משוב למערכת הבקרה על המיקום הנוכחי בו הוא נמצא.



### עקרון פעולה של מנוע SERVO

מנוע SERVO מופעל בדרך כלל על ידי **מיקום בקר** הגורם לו להגיע למיקום רצוי. פקודת התנועה מתקבלת בקר מטורמת למתח PWM שMOVEDר ליחידת הבקרה הנמצאת בתוך המנוע. ערך זה הוא **הערך הרצוי** של המנוע.

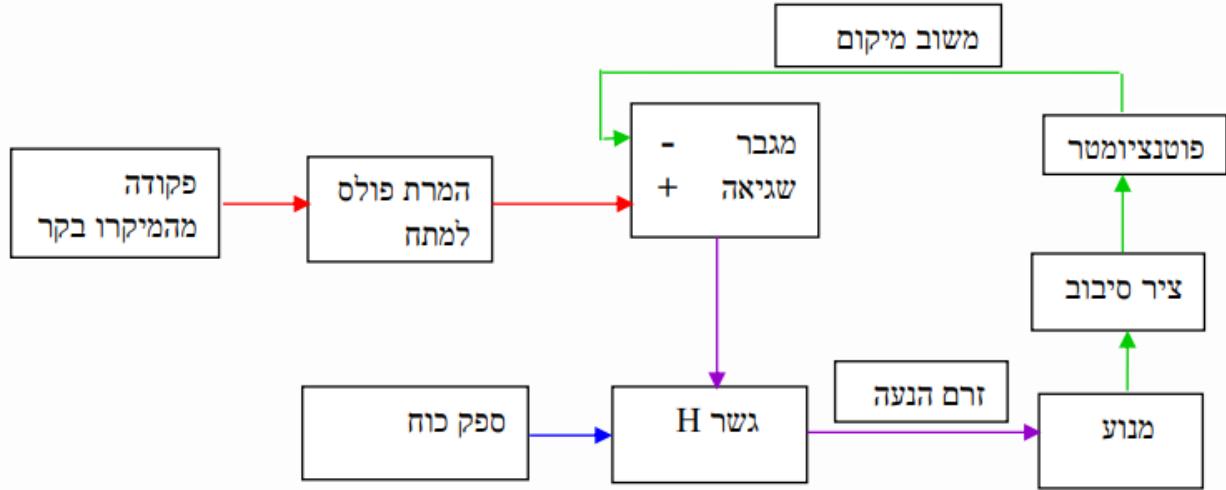
הערך המצוין של המנוע, נמדד בעזרת **פוטנציאומטר** המחבר לציר המנוע (הציר שיוצא מחלקו העליון של המנוע ואליו מחברים את המנגנון שיש להניע). סיבוב הפוטנציאומטר גורם לשינוי התחנכותות שלו ולשימושים של שינוי מתח המתח על רגלו של הפוטנציאומטר. מתח זה הוא

**הערך המצוין** שימושי לברkr הסגור. כך נוצר **שי פרש** נקרא מתח שגיאה (בין המתח המצוין ולבין המתח הרצוי). מנוע ה-DC-SERVO, יושיק לנوع בכיוון המתאים.

סיבוב מנוע ה-DC מתרחש יחד עם **סיבוב גלגלי השיניים** - שמהווים תמסורת מפחיתה) מאטת את מהירות הסיבוב ומגדילה את מומנט הכוח. (גלגלי השיניים מניעים את ציר היציאה ואת הפוטנציאומטר. ברגע שהפרש המתחים יהיה קטן מערך סף (קרוב ל-0), ייחידת הבקרה של המנוע תנתק מתח הנחוץ למשוך ה-DC והוא יחולן לנوع. עצירת המנוע תהיה בדיקות בזווית שנשלחה מהתוכנית בマイקרו בקר'.

אם התוכנית שולחת למשוך פקודה לנوع לזוויות מעבר ל-180 מעלות, המנוע יתחיל לרעוז מכיוון שהוא ינסה לsegor את השגיאה בין **הערך הרצוי** למוציא ולא יצליה. כמו כן יש ציר יציאה שמסומם מכני מהמנוע לעبور את זווית הסיבוב של 180 מעלות.

### נראה בעת כיצד נראה מערכת הבקרה של מנוע SERVO



מעולה, הנה גם הטקסט מהתמונה השלישית, מוקן להעתקה לוורד :

בצד שמאל מגיעו אוט הפקודה מהמיקרו בקר על הזרווית הרצואה שנקרא **הערך הרצוי**. זהו פולס עם **Duty Cycle** כאשר הזמן בין ה-ON ל-OFF קובע את הזרווית הרצואה. אותן זה נקרא גם **מיקום מטרה**.

معالג "מרת פולס למתח" מעביר למוגבר השגיאה מתח שמצוין מה-**Duty Cycle** של אוט הפקודה.

מוגבר השגיאה מקבל מהפוטנציאומטר מתח היחסי למיקום ציר המנוע, מתח זה הוא **המיקום הנוכחי של המנוע** ונקרא **ערך מצוי**.

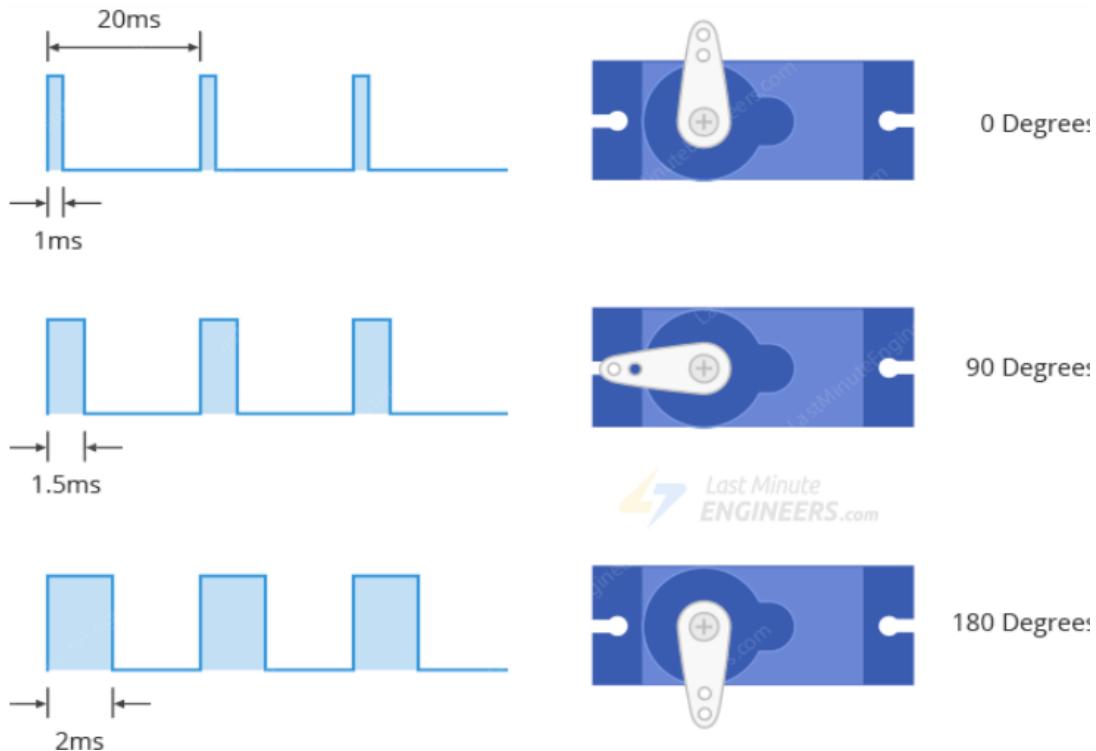
מוגבר השגיאה מגביר את **הפרש המתח** בין הרצוי למתח המציאות. יציאת המוגבר מחוברת לגשר - **H Bridge** שמעביר זרם המסובב את המנוע.

כאשר המנוע מסתובב הוא **מסובב את הפוטנציאומטר** שמעביר שוב מתח שלילי אל **הכניסה המהפקת** (המינוס של המוגבר) של המוגבר. ככל שהמנוע מסתובב וציר המנוע מתקרב אל הזרווית הרצואה  **הפרש המתח הולך וקטן**. כאשר מגיעים לזרווית הרצואה,  **הפרש המתח למוגבר הוא 0** ואין איינו מוציא פקודה לסייע המנוע והמנוע עצמן.

## כיצד עובד מנוע סרוו?

### עבודה עם פולסים:

אפשר לשלוט על מנוע SERVO על ידי שליחת סדרה של פולסים אליו. מנוע SERVO טיפוסי מחייב לפולס כל 20 אלףות השנייה (כלומר, תדר האות שנשלחה למיקרו בקר צריך להיות 50HZ). רוחב הדופק קובע את מיקום המנוע - SERVO. האירור הבא מראה את הזווית של המנוע עבור רוחבי שונים.



מיקום המנוע כתלות ברוחב הדופק של ה-DUTY CYCLE.

### מתיאור רוחב הפולס:

- **פולס קצר** ברוחב של 1 אלףות שנייה או פחות מסובב את מנוע הסervo ל-0-מעלות.
- **פולס ברוחב של 1.5 אלףות שנייה** מסובב את מנוע הסervo ל-90-מעלות.
- **פולס ברוחב של 2 אלףות שנייה** בערך מסובב את מנוע הסervo ל-180-מעלות.

**פולסים בטוחה של 1 ms עד 2 ms יסובבו את servo למיקום פרופורציוני לרוחב הפעימה.**

יש מנועי servo המסוגלים להסתובב 360 מעלות. רוחב הפולס במנועים אלו יכול לתת את הזווית הבאות:

- 0.5 ms נוthen זווית של 0 מעלות.
- 1 ms נוthen זווית של 45 מעלות.
- 1.5 ms נוthen זווית של 90 מעלות.
- 2 ms נוthen זווית של 135 מעלות.
- 2.5 ms נוthen זווית של 180 מעלות.

**הערה:** אין תאום מדויק בין רוחב הפולסים למיקום ולכון יהיה علينا לשנות את התכונות שלנו כדי להתאים לטוווח של מנוע-SERVO שלנו. כמו כן, **משך/רוחב הדופק** יכול להשנות בין יצרני מנועי SERVO שונים; לדוגמה, זה יכול להיות 0.5ms עד 2.5 ms עבור 180 מעלות ו- 1ms עבור 0 מעלות.

#### הטבלה הבאה מאפיינת את המנוע SERVO:

| טיואר   | עברית            | אנגלית                |
|---|------------------|-----------------------|
| 23.2 x 12.5 x 22 mm                                   | מידות            | Dimensions            |
| 9 g   | משקל             | Weight                |
| 0.12sec/60degrees (4.8V)<br>0.10sec/60degrees (6V)    | פעולה מהירות     | Operating Speed       |
| 1.3kg.cm/18.09oz.in (4.8V)<br>1.5kg.cm/20.86oz.in(6V) | מומנט מעכב       | Stall Torque          |
| 4.8V~6V   | מתח פעולה        | Operating Voltage     |
| Analog  | מערכת בקרה       | Control System        |
| CCW   | כיוון            | Direction             |
| 120degrees  | זווית פעולה      | Operating Angle       |
| 900us-2100us  | הפולס הנדרש      | Required Pulse        |
| None  | סוג מיסב         | Bearing Type          |
| Metal   | סוג גלגל השיניים | Gear Type             |
| Plastic   | סוג המנוע        | Motor Type            |
| 20 cm   | אורך חוטי החיבור | Connector Wire Length |

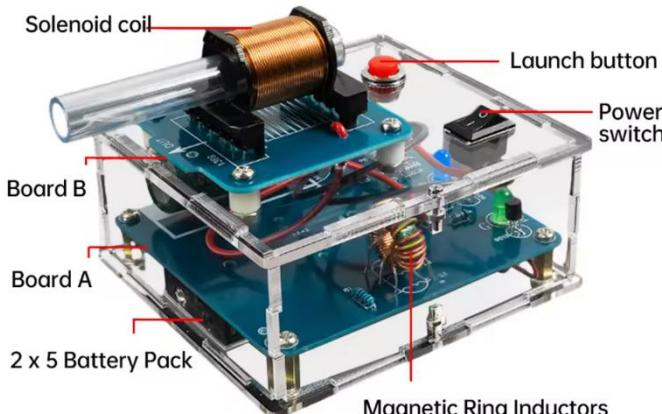
#### עבודה עם מנוע סרו ומיקרו בקר ESP32:

העבודה עם מיקרו בקר ESP32 בסביבת AEDUINO IDE דומה מאוד לעבודה עם כרטיסי מיקרו בקר ארדואינו (אונו, נאנו, מגה וכו').

התכונות כמעט זהה לתכונות עבור הארדואינו.

**הערה חשובה:** יש לשים לב כי המתח של מנוע ה-SERVO הוא בין 4.8V עד 6V ולא את מתח 3.3V שברכיבים.

## רובה אלקטرومגנטי (Electromagnetic gun)



### **תיאור כללי:**

המערכת המתוארת בתמונה היא רובה אלקטرومגנטי חד-שלבי הפועל באמצעות שדה מגנטי שנוצר ע"י סליל (Solenoid Coil), המפעיל גוף מתכתי דרך קנה פלסטי. הרובה נשלט ידנית באמצעות כפתור שיגור, ומתקבל אספקת החשמל ממארז סוללות פנימי.

### **מטרת המערכת:**

שיגור גופים מתכתיים גליליים קטנים (הנקראים כאן "Bombs") באמצעות כוח מגנטי – לצורכי הדגמה פיזיקלית, חינוכית או ניסיונית.

### טבלת הסבר של התמונה:

#### **רכיב**

#### **תיאור**

**Solenoid Coil סליל**  
**אלקטромגנטי**

סליל נחושת דרכו עובר זרם גבוה. יוצר שדה מגנטי חזק ברגע הפעלה ומושך את הגוף המתכתי לתוךו ב מהירות גבוהה.

**Launch Button כפתור**  
**שיגור**

**Power Switch מתג הפעלה**

מתג הפעלה/כיבוי ראשי של המערכת. מנתק ומחבר את אספקת המתח לכל המעגלים.

**Board B לוח עליון**

מכיל את הסליל, קונקטורים, כפתור ההפעלה, נורות בקרה, והחיבורים הפיזיים.

**Board A לוח תחתון**

מכיל את הרכיבים החשמליים המרכזיים – קבלים, נגדים, מיצבים, לולאות השראה וכו'. אחראי על ויסות המתח וההגנה על המערכת.

**2x5 Battery Pack מארז סוללות**

שתי סדרות של 5 סוללות המספקות מתח כולל גבוה (לרוב 18–24 וולט) אחראי על אספקת האנרגיה לסליל.

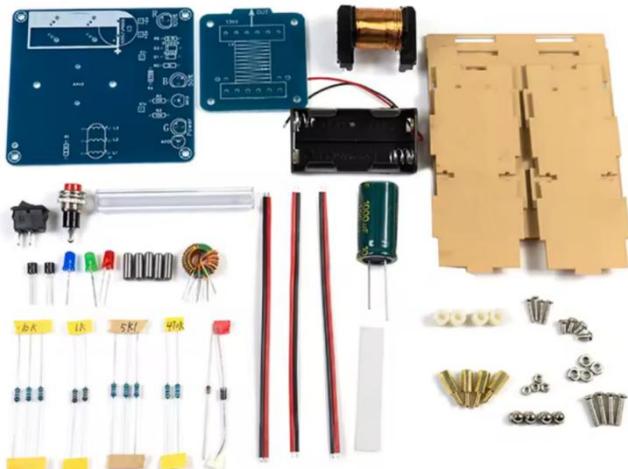
**Magnetic Ring Inductors**  
**חיישוקי השראה מגנטיים**

משמשים להחלהת הזרם, הפחיתה רעשים אלקטромגנטיים, ואחסון אנרגיה זמנית בمعالג.

**Bombs כדוריות מתכתיים**

גופים גליליים מתכתיים אשר מוכנסים לקנה. בזמן שיגור הם נמשכים דרך הסליל ויוצאים ב מהירות מהקצתה השנייה.

## תמונה של הרכיבים ברובה האלקטרומגנטי:



### עקרון פעולה:

- .1. המשמש מפעיל את מתג ההפעלה.
- .2. בלחיצה על כפתור השיגור, זרם מוזרם מהסולולות לסולנויד.
- .3. נוצר שדה מגנטי חזק בתוך הסליל.
- .4. גוף המתכת נמשך פנימה וሞץ לאורך הקנה.
- .5. לאחר מכן, הגוף י יצא במהירות מהקנה – ללא חלקיים נעים מכניים.

הטבלה הבאה מציגה את כל רכיבי הערכה לרובה האלקטרומגנטי כפי שמופיעים בתמונה לעיל, כולל תיאור תפקודיו של כל רכיב המערכת:

| רכיב                               | תיאור תפקודי  |
|------------------------------------|---|
| <b>לוח PCB ראשי</b>                | לוח המעל המרכזית שליטה מולחמים כל רכיבי האלקטרוניקה, כולל נגדים, קבלים וסלילים. |
| <b>לוח PCB שני</b>                 | לוח עזר – לרוב משמש לקישוריות בין סליל הירוי לבין הלוח הראשי או כקונקטור.       |
| <b>סולנויד (Solenoid Coil)</b>     | סליל נחושת דרכו עובר זרם ליצור שדה מגנטי חזק המאיץ את הקליע לאורך הקנה.         |
| <b>מחזיק סולולות (2x5)</b>         | מחזיק שני סטים של 5 סולולות – AA מספק מתח גבוה (לרוב 1.5V–24V) להפעלת הסולנויד. |
| <b>פלטות מארז (Housing Panels)</b> | חלקי הגוף המערכת – חיתוך לייזר מחומר מוקשה / (MDF) אקריליק (להרכבת המארז).      |
| <b>כפתור שיגור (Launch Button)</b> | כפתור אדום להפעלת השיגור – מזרים זרם מיידי לסליל לצורך ירי.                     |
| <b>מתג הפעלה (Power Switch)</b>    | mpsok הראשי להפעלה או כיבוי המתח הכללי של המערכת.                               |
| <b>נורת לד</b>                     | נורת חיוי – מדיליקה כשהמערכת פועלה או מוכנה לשיגור.                             |
| <b>צינור פלסטיק (קנה)</b>          | משמש כמיסילה לירוי – דרכו עובר הקליע במהלך השיגור.                              |
| <b>קליעים מתכתיים (Bombs)</b>      | גופים גליליים ממתכת אשר נמשכים אל תוך הסליל ויוצאים במהירות – משמשים להדגמה.    |

|                                    |   |
|------------------------------------|---|
| <b>טורואידים / סילילים טבעתיים</b> | רכיבי השראה מגנטיים לאגירת אנרגיה ולSHIPOR יציבות זרם.                            |
| <b>קבל אלקטROLITY</b>              | משמש לאגירת אנרגיה קצחה והעברת זרם חזק מיידי בעת השיגור.                          |
| <b>נגדים (Resistors)</b>           | רכיבים להגבלת זרם, קביעת מתח, או תיאום בין רכיבים שונים במעגל.                    |
| <b>חותמים ומוליכים</b>             | חותמים אדומיים/שחורים ל קישוריות חשמלית בין הלוחות, הסוללות, הסולנואיד והכפטורים. |
| <b>ברגים, אומים ומרוחחים</b>       | חלקים מכניים להרכבת הלוחות, קיבוע הסוללה, בניית המסגרת וחיבור כללי של המערכת.     |

### יתרונות המערכת:

#### **יתרון**

**שימוש בטכנולוגיה אלקטرومגנטית מתקדמת**

מודגים את עקרונות ההנעה המגנטית ללא צורך במכאניקה מסובכת.

**אין חלקים נעים מכניים  
פעול פשוט**

מקטין בלבד, שקט יותר וב吐וח יותר (יחסית).  
כפטור אחד לשיגור – מותאים ללמידה, הדוגמה וניסויים.

**גודל קומפקטי ונידות**

ניתן להפעיל את המערכת בכל מקום ולשאת אותה בקלות.

**רב פעימות**

ניתן לשגר מספר פעמים עם אותו "פיצוץ".

#### **הסבר**

#### **הסבר**

**עוצמת שיגור מוגבלת**

בשל מגבלות מתח, סליל יחיד וגוף קל – העוצמה וה מהירות יחסית נמוכות.

**יעילות אנרגטית נמוכה**

חלק גדול מהאנרגיה מתבזבז כחום ולא תורם להאצת הגוף.

**חימום רכיבים**

שימוש רצוף עלול לגרום להתחממות יתר ולפגיעה במעגלים.

**אין בקרת תזמון מתקדמת**

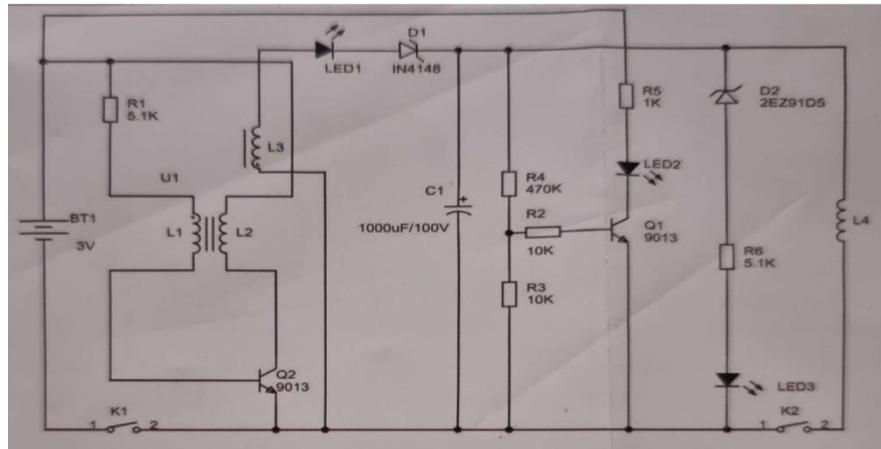
לא חייני מיקום – אין ניתוק מדויק של השדה ולכן תיתכן האטה בשלב מסוים.

**טעינה איטית (אם משתמש בקבלים)**

אם קיימים שלב טעינת קבלים, יידרש זמן בין שיגור לשיגור.

רובה אלקטرومוגנטית הוא מערכת מרתקת ומשולבת – מגדים פיזיקה, חשמל, אלקטרוניקה ומכאניקה ברמה מסוימת. הוא מהוות פלטפורמה חינוכית מצוינת להבנת עקרונות השראה מגנטית, בקרה, ואנרגיה. מבנהו הקומפקטי והשימוש הנגיש הופכים אותו לכלי לימוד ולמעבדה ניידת לכל דבר.

**הסביר על מעגל חשמלי - רובה אלקטرومגנטי**



הסבר כללי:

המגל משתמש ברכיבים כמו קבילים, סילילים (L) טרנזיסטורים, דיודות, וורות LED כדי לטעון את סוללה במתוך גבוח ואז לפroxו אותו דרך סליל אלקטромגנטי לצורך יצירת דחף (למשל, שיגור פרויקטיל מתקתי קטן).

הסבר לפי אзорים:

אזר הספק:

**K1**- מתג הפעלה ראשי שמחבר את החסמל.  
**BT1**- סוללה (V)3- מספקת את האנרגיה למעגל.

פרק טעינת הקבל:

- C1** – קבל  $\mu\text{F}$  1000–100 נטען בהזרגה דרך המעלג, ואוגר אנרגיה שתשוחרר בבת אחת לשיגור.
- D1** – דיודה 1 – (N4148) מגינה על המעלג מזרם חוזר.
- LED1** – מצין שהקבל נתען.
- L1, L2, L3** – סילילים שנראים כחלק ממעלג מתנדן/רוזוננס שמעביר אנרגיה לקבל.

חלק בקרה:

**Q1, Q2** - טרנויסטוריים מסוג 9013 – מפעילים או חוסמים את מעבר הזרם.  
**R1-R6** - נגדים – קובעים את זרימת הזרם לבסיס הטרנויסטוריים, וכן שולטים עליהם.  
**LED2** - חיוני נוסף למצב ההפעלה של השלב הבא.

## **חלק הפעלת סליל השיגור:**

- D2** -**Zener Diode** או – SCR מפעילה שחרור מהיר של האנרגיה מהקובל.
- L4** -**סליל השיגור** – כאשר הקובל נפרק דרכו, נוצר שדה מגנטי חזק שיכול לדחוף אובייקט מתחתי קטן.
- K2** -**מתג הפעלה לשיגור**.
- LED3** -**נדלק** באשר מהתבצע השיגור.

## טבלת רכיבים:

| <b>רכיב</b> | <b>תיאור</b>            | <b>תפקיד</b>           |
|-------------|-------------------------|------------------------|
| BT1         | סוללה 3V                | מקור מתח               |
| K1, K2      | מתגים                   | K1-הפעלה, K2-шиגור     |
| Q1, Q2      | טרנזיסטורים 9013        | מגברים/מתגים לברכה     |
| C1          | קбл $\mu F$ 1000 - 100V | אגירת אנרגיה חשמלית    |
| D1          | דיודה N41481            | הגנה מזרם חוזר         |
| D2          | דיודה Zener / SCR       | פריקה פתאומית של הקבל  |
| R1-R6       | נדדים                   | קייעת זרמים לברכה      |
| LED1-LED3   | נוריות חיומי            | מראה מצבים פעולה שונים |
| L1-L4       | סלילים                  | יצוב, תדר, שיגור       |

### פירוט רכיבי המעגל ותפקידם האלקטרוני:

#### : C1 : קבל האגירה ( $\mu F$ -100V-1000)

- **תפקיד ראשי:** המאג'ר הראשי של האנרגיה הפוטנציאלית החשמלית. הקבל נתען באופן יזום למתח נבוה (עד 100V) ומחזיק את האנרגיה זו עד לשלב השיגור.
- **ניתוח מקצעי:**
  - **קיבול :** ( $C = 1000 \mu F$ ) הקיבול נמדד בפאראדים (Farads) וקובע את כמות המטען (Q) הנדרשת כדי להגיע למתח נתון.
  - **מתח נקוב :** (100V) זה המתח המקסימלי הבטיחותי שהקבל יכול לעמוד בו. במעגל דחף, חשוב שהמתח הנספג במהלך הטעינה לא יעלה על ערך זה.
- **אנרגייה אగורה :** האנרגיה הפוטנציאלית המקסימלית האגורה בקבל נתונה על ידי הנוסחה

$$E = \frac{1}{2} CV^2$$

עבור קבל זה בטעינה מלאה (100V) :

$$E = \frac{1}{2} \cdot (1000 \cdot 10^{-6} \text{ F}) \cdot (100 \text{ V})^2 = \frac{1}{2} \cdot 0.001 \cdot 10,000 = 5 \text{ Joules}$$

זהו אנרגיית השיגור המקסימלית העומדת לרשות המערכת.

## D1: דיוידת הגנה (1N4148)

- **תפקיד ראשי: חסימת זרם חוזר (Reverse Current Blocking).** הדיוידה מודדת שהזרם יזרום רק מהמעגל המגביר המעגל שמשתמש ב- (L1,L2,L3) אל הקבל (C1).
- **ניתוח מקצעי:**
  - **הגנה מפני פריקה:** בזמן השיגור (כאשר הקבל נפרק), המעגל חוויה שינוי דרמטי במתח וזרם גדול. הדיוידה מונעת מהזרם הפורק להזיק לרכיבי המגבר (כגון טרנזיסטורים) שעלוים להיות עדינים.
  - **דגם הדיוידה :** (1N4148) זהה לדיויד מיתוג מהירה וקטנה (Small Signal Switching Diode, עם זרם קדיימה מקסימלי נמוך יחסית (כ- 150mA) לעומת (1mA) לדוגמה, להניח שהיא ממוקמת **במעגל הטעינה בלבד**, היכן שהזרם קטן יחסית (שכן הקבל נתען בהדרגה, ולא חלק מעגל הפריקה של ה-J5-שים נדרש דיוידה חזקה בהרבה (כגון דיוידת UF4007 או FR107).

## נוירית חיוי LED1

- **תפקיד ראשי: אינדיקציה חזותית ל乓בוב הטעינה.**
- **ניתוח מקצעי:**
  - נוירית זו מעידה כי המתח על הקבל (V) הגיע לרמה מספקת או מלאה. היא מחוברת בדרך כלל **במקביל** לקבל בטור עם נגד הגבלת זרם מתאים. כיוון שמתהה הטעינה הוא 100V, יש צורך בנגד הספק גובה או במנגנון מיתוג (כגון נוירית ניאוון קטנה או טרנזיסטור) כדי להבטיח שהיא לא תישרף בגל המתהה הגבוה.

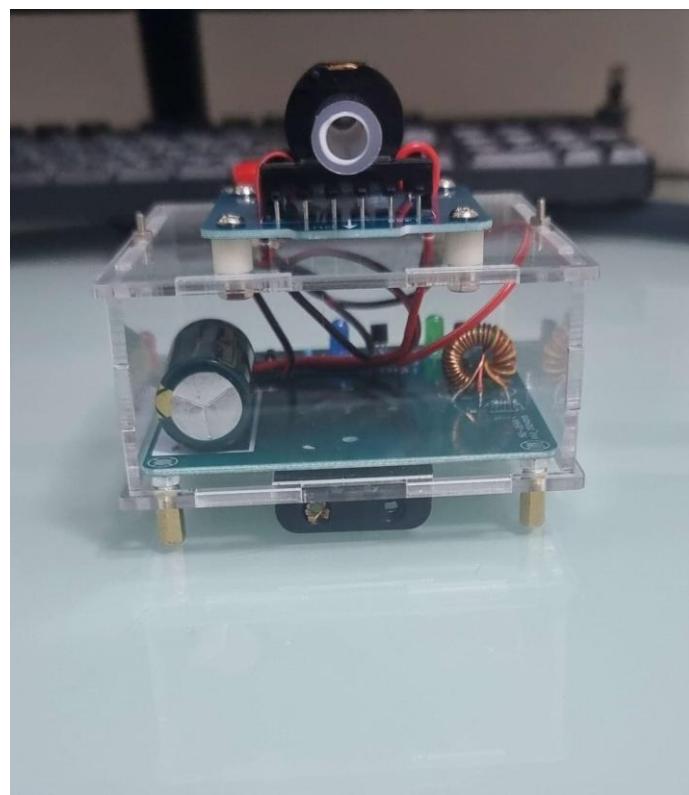
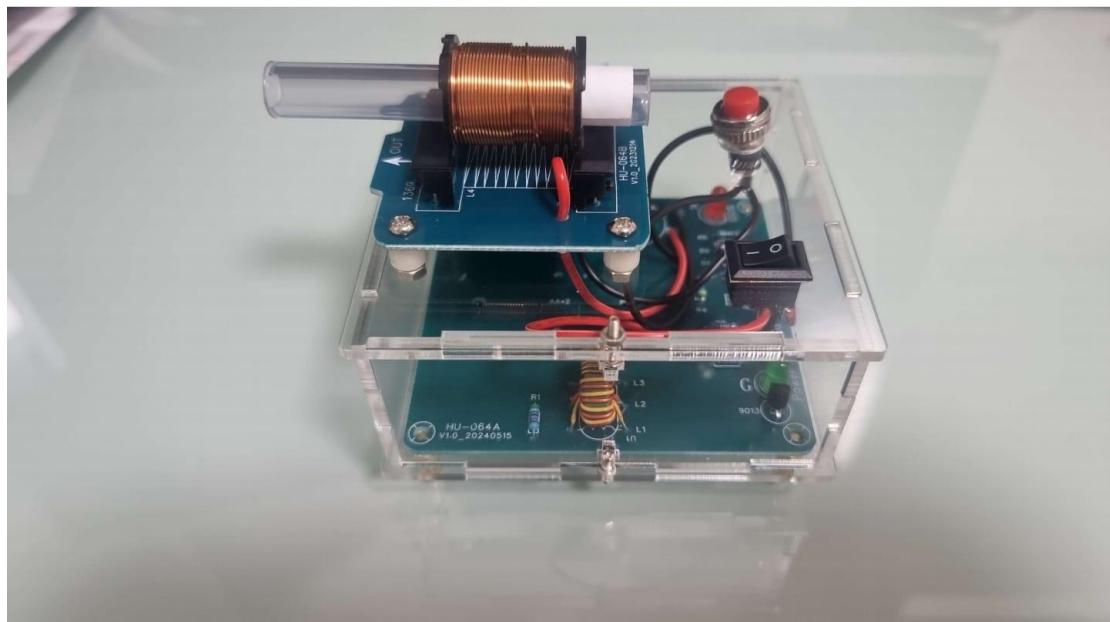
## L1, L2, L3 : סילילים (Inductors/Coils)

- **תפקיד ראשי: רכיבים אלו הם ליבת **מעגל המגביר** (Boost Converter) או **ממיר מתח מורם** (Flyback Converter). תפקידם הוא להפוך מתח כניסה נמוך (כגון סוללה של 12V – 3V למתח גובה של 100V)**
- **ניתוח מקצעי:**
  - **ממיר DC-DC :** סילילים אלה משמשים ליצור שדה מגנטי. על ידי מיתוג מהיר של זרם דרך סליל ראשוני (L1) או (L2) באמצעות טרנזיסטור, נוצרת השראה הדודית שמייצרת מתח גובה בסליל משני L2 או (L3).
  - **דחית זרם ישיר :** המעגל המגביר פועל על העיקרונו של **שינוי מהיר בשטף המגנטי**, שיוצר מתח גובה על פי חוק פאראדי. המתח הגובה הזה מיושר באמצעות דיוידה (אולטי, D1, אם כי D1 היא חלשה מדי, או דיוידה נוספת) ונשלח לטיענת C1.

## סיכום ומסקנה (הקשר המערבי):

- המערכת יכולה מתארת **מחולל דחף אלקטромגנטי**, העובר את שלבי הפעולה הבאים :
1. **הגברה :** המתח הנמוך מוגבר ל- 100V באמצעות מעגל המתנד/הממיר (L1,L2,L3).
  2. **טעינה :** המתח המוגבר עובר דרך דיוידת הגנה (D1) ונטען לתוך **קבל האנרגיה** (C1).
  3. **МОכנים :** כאשר C1 מגיע ל- 100V (אוגר J5), נוירית החיווי (LED1) נדלקת.
  4. **שיגור :** בשלב זה, באמצעות מתג מיוחד שאינו מצוי בראשימה, כגון טרנזיסטור SCR או MOSFET בעל זרם גבוה), משוחררת האנרגיה האגורה ב- C1 **בבת אחת** לתוך סליל שיגור (העומס), מה שמייצרת את הדחף האלקטרומגנטי הדרוש לשיגור.

תמונה של התוצאה הסופית של הרובה:



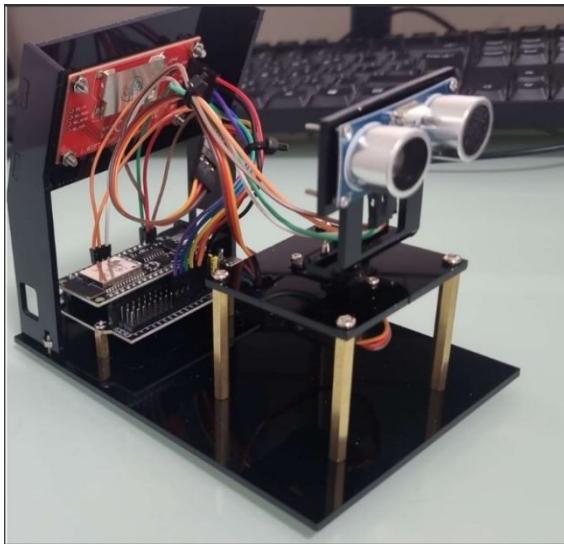
קישור לסרטון בו אני מרכיב את המערכת של הרובה האלקטרומגנטי ומדגים את אופן פעולתו  
וכיצד מתבצעת הירייה (יש להחזיר CTRL + על קישור של הסרטון)

<https://youtu.be/5IMt-HXeClk?feature=shared>

## מיני רdar (Mini Radar)

### המטרה והעקרון:

המטרה של הפרויקט הייתה ליצור מערכת שתסורך את הסביבה הקרובה שלה, תזהה מכשולים ותציג אותם על גבי מסך, בדיקות כמו מכ"ם אמיתי.



### הרכיבים והתפקידים שלהם:

כמו בכל פרויקט מוצלח, גם המערכת זו מורכבת מכמה חלקים שפועלים יחד :

- המוח : זהו המיקרו-בקר, הלב והמוח של המערכת. הוא שולט בכל הרכיבים ומבצע את החישובים המורכבים בזמן אמיתי.
- העיניים : החישון האולטרה-סוני, שמצויר קצר שתי עיניים קטנות. עין אחת משדרת פולס קולי שאנחנו לא יכולים לשמעו, והשנייה קולטת את ה聲 שזור. הנתון הקרייטי שאנחנו מקבלים הוא הזמן שלקח לפולס הזה לצאת ולהזוז.
- הזרוע : זהו מנוע הסרבו. הוא מאפשר לחישון שלנו לנوع מצד לצד, מה שנutanן למערכת את היכולת לסרוק שטח שלם ולא רק נקודה אחת.
- התוכנה : כל החלקים הפיזיים לא שווים כלום בלי הקוד שכתבתי. התוכנה היא זו שמנחה את מנוע הסרבו לנوع בזווית שונות (למשל, 5 מעליות בכל פעם), ובכל עירה היא מורה לחישון למדוד את המרחק לעצם. לאחר מכן, הקוד שולח את נתוני המרחק והזווית למסך חיצוני או למחשב.



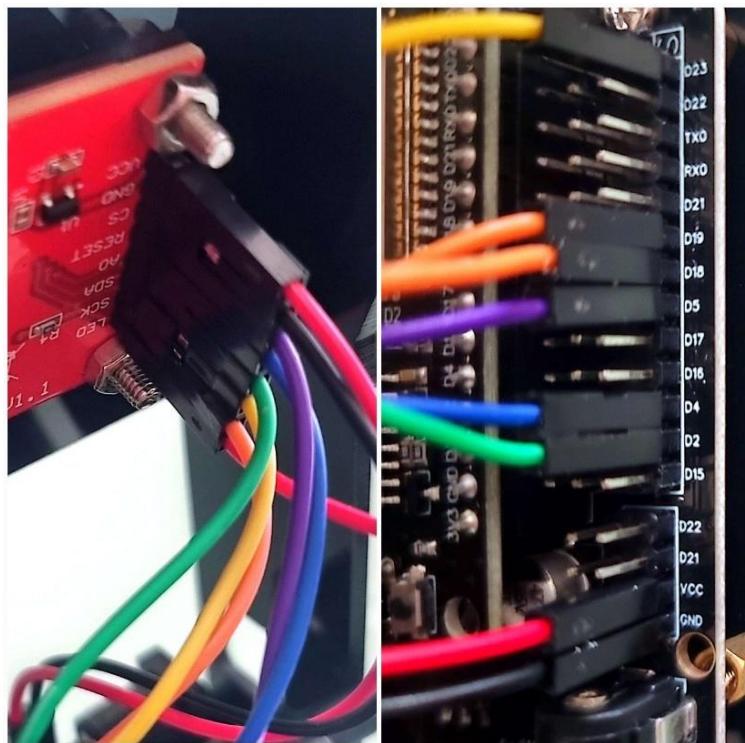
## **איך זה עובד בפועל?**

התהליך פשוט ואלגנטי: המנוע מסתובב בזווית מסוימת, החישון מודד את המרחק לעצם הקרוב מזונים לקוד. התוכנה לוקחת את הנתונים – **הזווית והמרחק** – ביותר, ושני הנתונים האלו ומציגו אוטומטית ויזואלית על מסך. ככל שהמכשול קרוב יותר, כך הנקודה שתופיע על ה"מכוון" תהייה קרובה יותר למרחוק. על ידי חזרה על התהליך בכל זווית לאורך הקשת של 180 מעלות, אנחנו מקבלים תמונה שלמה של הסביבה שנמצאת מולנו.

הפרויקט הזה מדגים שימוש פשטוט של רכיבים יכול ליצור מערכת מורכבת ויעילה, והוא פותח את הדלת ליישומים מרחוקים כמו רובוטים אוטונומיים, מערכות אבטחה ביתית ועוד.

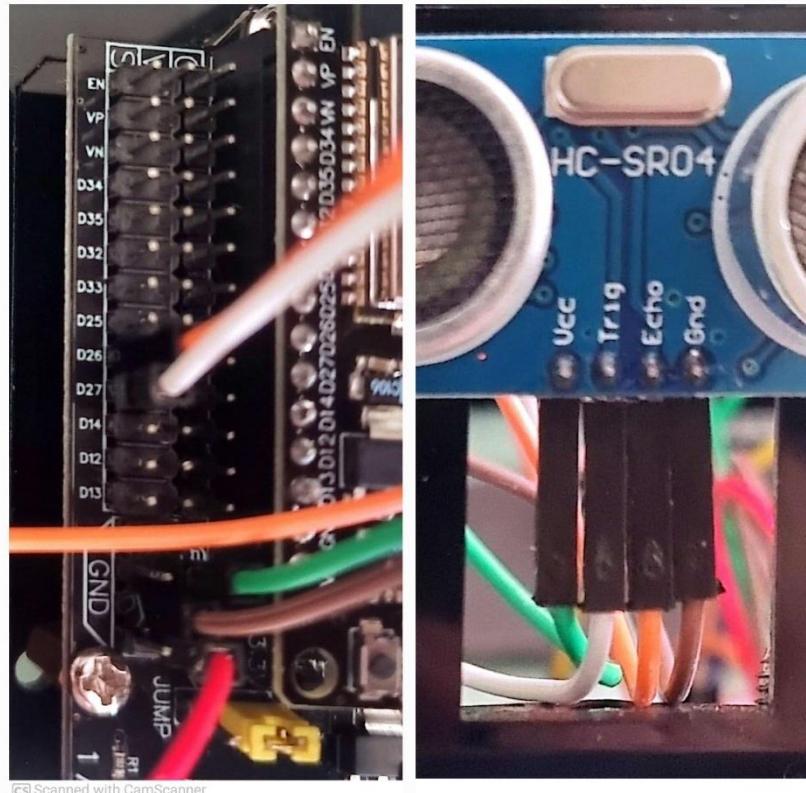
**נראה את אופן חיבורם של המძק והחישון האולטרסוני אל המיקרו בקר בזורה ברורה ומקורה יוטר:**

[המסך הקטן מחובר באופן הבא:](#)



- ניתן לראות ש-VCC במסך מחובר ל-VCC שבמיקרו בקר.
- היפין GND שבמסך מחובר ל-GND שבמיקרו בקר.
- רgel CS שמוצגת במסך מחוברת לפין D5 שבמיקרו בקר.
- רgel ה-RESET שבמסך שלגנו מחוברת לפין D4 שבמיקרו בקר.
- רgel 5V יכולה להיות מחוברת לרgel אנלוגית או דיגיטליית תליי בפקודות שנרצה להשתמש, אני חיבורתי רgel זאת לרgel דיגיטלי/Sifرتית D2 שבמיקרו בקר.
- רgel SDA שהיא למעשה הרgel SDI לאחר משום שהוא מתקשר עם המיקרו בקר דרך תקשורת טורית SPI, ישנו בלבול משום שהוא רגליים שרגל SDA שימושית בתקשורת I2C אך יבואנים סיניים החלו לייצר מסכים ולרשום גם במסכים המתקשרים דרכם תקשורת SPI את השם של הרgel SDA והוא מאפיינית את התקשרות שבין הממסך למיקרו בקר.
- רgel SCK מחוברת לפין D18 (החוות השני הוא של המנוע סרבו שמחובר לפין D19).
- למעשה רgel SCK זהה הרgel שמעבירה את אותן הש�ון.
- רgel ה-LED מחוברת ל-3.3V בצד השני של המיקרו בקר.

נראה את החיבור של החיבורן האולטרסונייק:



- רgel-h-TRIG מחוברת לפין D27 שבמיקרו בקר.
- רgel-h-ECHO מחוברת לפין D26 שבמיקרו בקר.
- המתח VCC שבхиישן מתחבר לפין 5V שבמיקרו בקר משומש לחיזיון זה בדומה למונע סרווו שלנו עובדים עם 5 וולט והמיקרו בקר ESP32 עובד על 3.3V.
- הארקה GND מחובר כמובן לאזמה GND שבמיקרו בקר שלנו.

**קישור לסרטון בו אני מרכיב את המערכת של המיני רадאר. (יש ללחוץ Ctrl ואו על הקישור)**

<https://youtu.be/TOxmNhN4LiU>

## תיעודים-בדיקות

### תיעוד של אופן פעולה החישון האולטרסוני ומדידת מתחים 5.9.25:

בתיעוד זה נוכל לראות את אופן פעולה החישון האולטרסוני, מדדתי את המתח שהספק מקבל מהמיקרו בקר ESP32 ובנוסף בדקתי כמה מתח המיקרו בקר מוציא מהפין של ה- 3.3V האם הוא באמת מוציא מתח זה או שלא, בבדיקות קיבلت שắcל תקין וחישון קיבל טיפה פחותה מתח 5 וולט וזה עלול להיגרם ממספר סיבות הבאות :

מתח ה- USB הסטנדרטי מוגדר להיות 5.0V, אך יש לו **סבירות** (Tolerance). קריאה של 4.5V נובעת ככל הנראה מהשילוב של הגורמים הבאים :

#### 1. מפל מתח בכבול ה-

זהו הגורם העיקרי :

- **התנגדות הכבול**: לכל כבל, גם הקצר והaicotti ביותר, יש התנגדות חשמלית. כאשר ה- ESP32 מחיבורים וဓוריים זרים (ה- **העומס**) נוצר **מפל מתח** לאורך הכבול, לפי חוק אוהם :  $V_{drop} = I \times R$ .
- **כבלי USB דקים וארכיים**: כבלים דקים יותר (בעלי מס' AWG גובה יותר, כמו AWG 28 במקום AWG 20 עבה יותר) או ארוכים, הם בעלי התנגדות גבוהה יותר, מה שմגביר את מפל המתח.
- **הקריאה שלך נушה "תחת עומס**": אם מזגת 4.5V בזמן שהוא ESP32, חישון האולטרסוני ומצלמה Wi-Fi שלו פועל, המתח יורד עקב צריכת הזרם.

#### 2. סבירות תקן ה-USB:

תקן USB 2.0 מאפשר למתח לרדת עד 4.4V בחיבור ל"Hub Port" (SKU פחות חזק) או לכל הפחות 4.75V בחיבור לשקע ורגיל, עד לצריכה המקסימלית המותרת. ערך של 4.5V נופל בטוויה הסביר, במיוחד אם אתה משתמש בכבול ארוך או ביציאת USB של מחשב נייד או רכזת(Hub).

#### 3. מפל מתח ברכיבי לוח ה-ESP32:

המתח עבר דרך מספר רכיבים על לוח ה- ESP32 שלך :

- **הגנת זרם יתר** (Over Current Protection): להזנת רבים מכילים נתיך הנitin לאיפוס או רכיב ההגנה אחר שמוסיף התנגדות קטנה וגורם למפל מתח קל.
- **דיודת הגנה** (Protection Diode): דיודה הגנה בכניסה ה-USB- יכולה לגרום למפל מתח של כ- 0.2V עד 0.7V תלוי בסוג.

לסיום, הקריאה של 4.5V היא תוצאה של הזרם שצורך ה- ESP32 (והחישונים המתחברים אליו) בשילוב ההתנגדות של כבל ה- USB ורכיבים שעל הלוח. אם הלוח עובד יציב, המתח הזה **תקין לחלוtin עבורה**.

את המדידות שיצאו ניתנו לראות בהסבר על החישון האולטרסוני במס' מסביר על החישון ואופן פועלתו בסוף ההסברים הנסתית תחת הכותרת "מדידות מתח ואופן פעולה" מצרך כאן סרטון מיפוי בו אתיודתי את אופן הפעולה ואת המדידות וכמוון אכנים את הסרטון לאתר שלי תחת כתורת בדיקות.

קישור הסרטון : <https://youtu.be/qJxPFXFTNuq>

## תיעוד של אופן הפעולה 20.9.25:



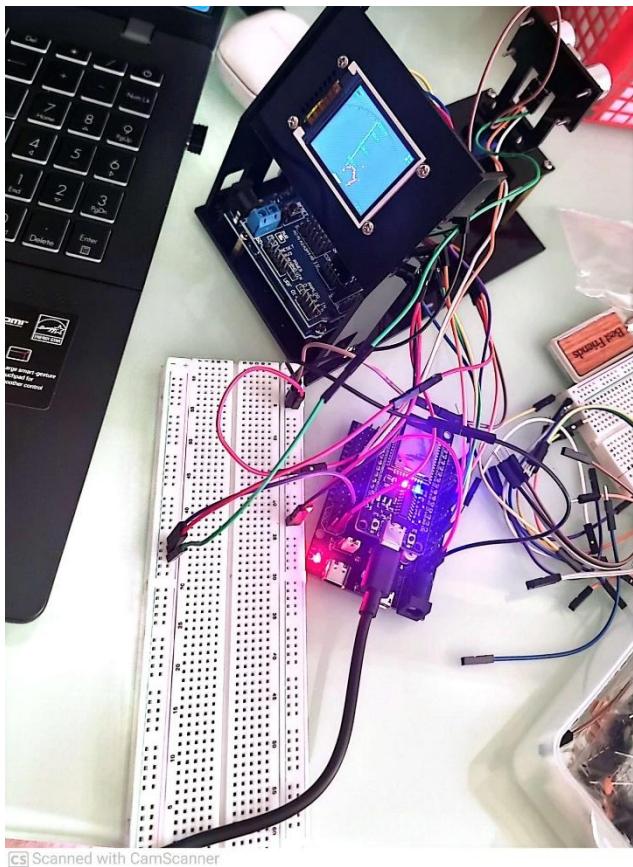
Scanned with CamScanner

אחד הביעות הגדולות שיצא לי להתמודד בפרויקט זה היה להפעיל את מערכת הרdar.

נתקלתי בבעיות כמו למשל ספירות לא תואמות הגדרת פורטים לא נכונים. בעיות אלו פתרתי על ידי חקירה על הפונקציונליות של המיקרו בקר ובנוסף על בסיסו הפורט בעזרת הטרמינל במחשב, אלו היו הפוצדרות המשובכות אך המועלות ביותר מאחר והן לימדו אותי להתמודד עם בעיות כאלו לפעם הבאות במידה ואתקל בהן.

נוסף על כך אחד הביעות שיצא לי להתמודד היה הפינים שהוגדרו להיות "בעיתיים" לשימוש, הכוונה שבמיקרו בקר אי אס פין פינים בהם כדי להשתמש מושם שנוחים יותר עבר פעולות מסויימות ויישם כאלו שמדובר בעובדה ספציפית יותר. לדוגמה יכול לומר שהמנוע שלי לא עבד כאשר חיבורתי אותו לפין 15 ואז שיחקנתי קצת עם הפינים ומתי שחברתי SERVO לפין 19 הוא התחליל לעבוד והתנסכרן בעובדה ייחד עם המסק והחישן אולטרה סוני שלמעשה נמצא עליו. בנוסף אחד הביעות הגדולות ביותר היה להפעיל את המסק. גרפיקה זה צעד גדול בפרויקט ויחסית מסוובך, לא כל הפונקציות תואמות למיקרו בקרים ספציפיים וצריך לנשות כמה פונקציות על מנת למצוא את המתאימה ביותר עבור אופן הפעלה שלו.

마וחר ואני יוצר תמונה הרdar שפועל בזמן אמת אז עלי לא להשתמש רק בתצוגה רגילה אלא ממש בתצוגת עיצוב גרפי חדשני ממעני להשתמש בפונקציות ייחודיות עבור פעולות אלו. בסופה של דבר עם קצת כישלונות ולמידה הצלחתית להבין יכון טעיתי בכל הניסיונות הללו ולמדתי מזה המונ.



Scanned with CamScanner

גם כאן בתמונה ניתן לראות את הניסיון הפעלה על מיקרו בקר אי אס פי.

בהתחלת חיבורתי הכל דרך מטריצה ולבסוף ארגנטית את זה על המערכת עצמה כדי לשפר את הנראות והנוחות במהלך השימוש בה.

בתמונה קצת קשה לראות את הרדאר שנוצר על המסך וכמובן שהתמונה לא משדרת את התזוזה של המנוע סרוו ואת הקליטה של החישון אולטרסוני אך כדי להמחיש זאת אני צירפתי לכטוטון שלי לבדוק את אופן הפעולה של המערכת.

בסרטון אציג את הבניה כולה של המערכת ואת אופן פועלתה בסוף הסרטון כדי שניתן יהיה לראות ולעקוב אחר השלבים צורת הבניה של המערכת.

בנוסף הסרטון יעלה לאתר שניתן לגשת אליו דרך ספר זה בקישור שאציג שייהי תחת הקותרת **"קישור לאתר"**

#### תיעוד של אופן פעולה הרדאר 21.9.25 :

בניסיון זה בדקתי את סימון הנקודות הצהובות שהן מופיעות עברו אובייקטים רחוקים מ-100 סנטימטר. הנקודות האדומות שופיעו על הרדאר אלו נקודות שמאפיינות אובייקטים הנמצאים בתחום של 0 עד 100 סנטימטרים מהחישון אולטרסוני.

תיעדתי ניסיון זה והוספתי סרטון שמראה כיצד בדקתי זאת הסרטון הופיע הפוך החלק של הבדיקה לקראת הסוף אבל זה לא פרק זמן ארוך וניתן להבחן בכל זאת את הנקודות הצהובות. אני מצרף כאן קישור לסרטון שלי שנמצא ביוטיוב וגם הכנסתי את הסרטון תחת הקטגוריה "בדיקות" בתוך האתר שיצרתי שיופיע בעמודים האחרונים של הספר פרויקט שלי.

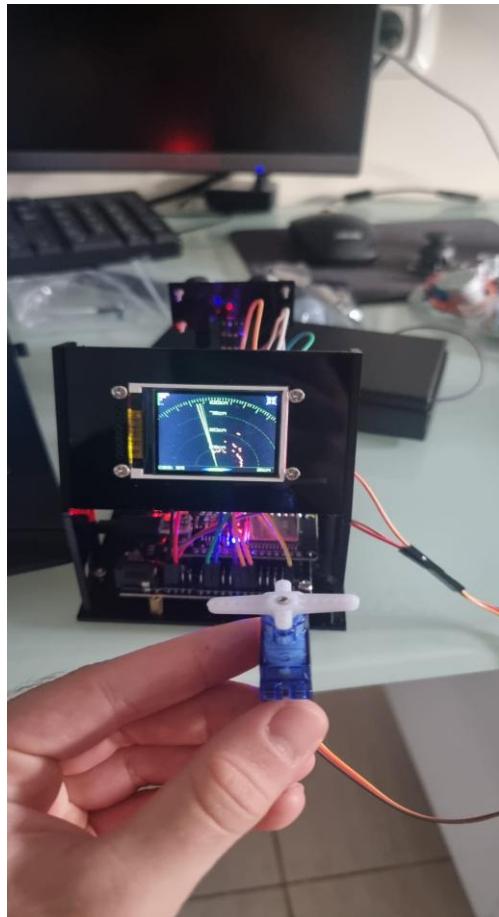
קישור לסרטון תיעוד של ניסיון יצירתי ובדיקה של הנקודות הצהובות שמנדריות אובייקטים שנמצאים למרחק מעל 100 סנטימטרים כולל מהחישון האולטרסוני:

[קישור לסרטון :](https://youtu.be/8BiuVFxUo20)

### **תיעוד של אופן הפעלה כאשר חיבורתי סדרו נוסף שיחיה אחראי על הסיבוב של התותח 23.9.25:**

בניסוי זה הקפנתי צעד יחסית מושם בפרויקט שלי משומש שהצלחתו לבצע סנכרון בין שני המנועי SERVO שיש לי במערכת שלי. ה-SERVO הראשון אחראי על לסובב את החישון האולטרסוני שסורק את הסביבה בטוחה של 180 מעלות החל מ-0 מעלות ועד ל-180 מעלות. המנוע ה-SERVO השני שלי אחראי למשעה על הסיבוב של המשטח עליו לחבר את הרובה האלקטרומגנטי שלי. ברגע שמתגלו נקודות אדומות שבמשמעון שהאובייקט נמצא בתוך של 100 סנטימטר ופחות פחות או המנוע ה-SERVO השני נכנס ישירות לפועל מהותו המוקם בו המנוע O SERVO הראשון שמסובב את החישון האולטרסוני נמצא מתחת וברגע שייתגלת שוב נקודה אדומה שמשמעותה על אובייקט או תבצעו ריריה מהרובה האלקטרומגנטי אך זה בשלבים מתקדים יותר כרגע בצעתי את הניסויון של המנוע ה-SERVO השני עם הראשון יחד.

**נראה תמונה ואסביר את החיבורים הנוספים שבעת למיקו בקר על מנת שהמנוע SERVO השני יתחל לסתובב:**



זאת למעשה התמונה בה אני מחזק את המנוע O SERVO השני, למעשה אי אפשר להבין שהמערכת אכן עובדת מהתמונה פשוטה וכן חשבתי איך סרטון ביוטיוב וקיים בתוך הספר וכמוון שאכנים את התיעוד לקטגוריות הבדיקות שבאתר שלי.

המנוע SERVO מחובר לרgel D25 שבמיקרו בקר ESP32 ואת שאר החוטים שבמנוע SERVO צירפתי את החוט האדום ל-GND ואת החוט החום חיבורו לאדמה.

בקוד כל מה שעלי היה להוסיף זה את ההגדשה של ה-SERVO הנוסף ולאיזו רגל הוא מחובר במיקרו בקר, ובנוסף לכך היה עלי להוסיף בתנועה הלווי וה坦ונה חזור בתוך התנאי AiFa שרשום המרחק קטן מ-100 שזה לעומת הנקודות אדמות מה שאומר שהאובייקט נמצא בטוחה של 100 סנטימטרים ומה בשנייהם היה עלי להוסיף פקודה שתפעיל את המנוע SERVO השני או במידה והמרחק של האובייקט גדול מ-100 אז התוינו של המנוע SERVO השני פוסקת והמנוע עצם.

**מציף קישור לסרטון בעמוד שלי ביוטיוב:**

<https://youtu.be/eeDQquCGN5I>

### תיעוד של אופן פועלות המסק טאץ' המשך השני בו אני משתמש בפרויקט 27.9.25

בתיעוד זה אציג את אופן פועלות המסק השני זהו מסק טאץ' בו משתמש כדי להראות את מצב התאורה והטמפרטורה ואוכל לעבור בין מצבים שונים שאחלייט לצרף על המסק.

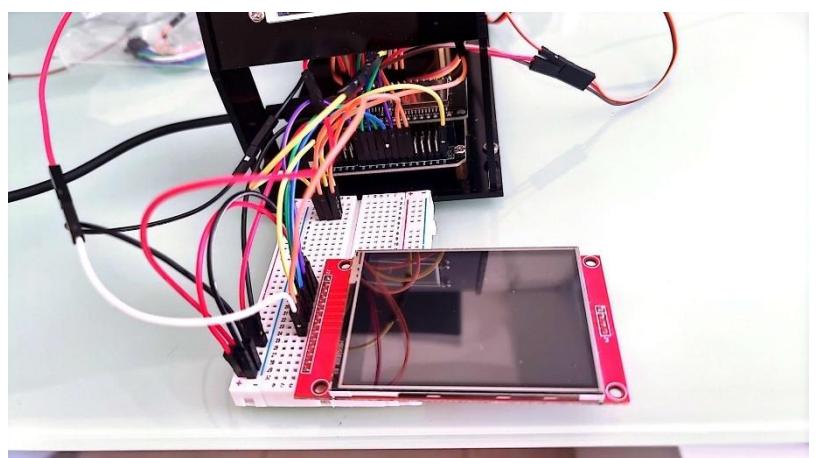
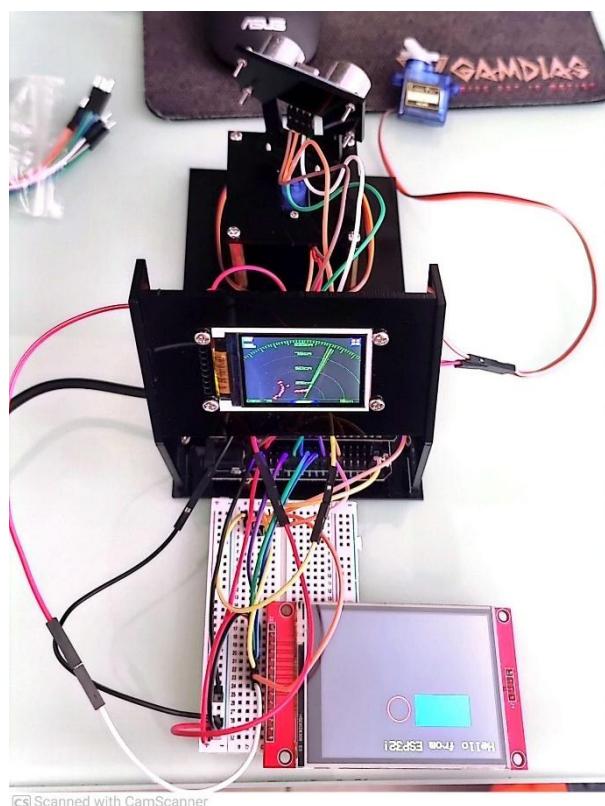
הקישים הבאים התמודדי עלי מנת לצרף את המסק השני עיביר ייחד עם פועלות המסק הראשון הינו בעיקר מאופיינים בהגדרת ספירות אחירות בין המסק שלו לא תמק, לאחר זמן ממושך וחקירה לגבי הנושא מצאתי את הספריות המתאימות שעליין הרחבותי והסבירתי בספר זה כאשר הסברתי על המסק טאץ'.

מצרף קישור לסרטון בו אני מבצע מדידת מתחים יחד עם אופן פועלות המסק השני. הקוד שצירפתי זהו קוד בסיסי שמדפיס את המשפט Hello from ESP32 ומתחתיו יש מלבן שכלי כחול ועיגול שהיקפו אדום והפניהם הוא במצב צבע (בצבע שחור). זהו אותו הקוד שהציגתי בהסבר על המסק טאץ' (המסק השני בפרויקט שלו), لكن ניתן לראותו בצורה מלאה וברורה יותר בספר זה תחת הכוורת והסבירים של "מסק טאץ' TFT 9341".

קישור לסרטון: <https://youtube.com/shorts/a2xaMos41Qk?feature=share>

### תיעוד של חיבור המסק השני למערכת כולה ואופן פועלות המערכת 28.9.25

כאן תיעדרתי את אופן החיבור של המסק השני למערכת כולה, המסק מציג בניסיון זה את אותו הדבר כמו שהראה בתיעוד הקודם אך ניסוי זה היה מדרגה משמעותית יותר בהתקומות הפרויקט אציג את אופן פועלות המערכת בסרטון ניסף וכמובן אציג מספר תמונות כדי שנייתן יהיה לראות זאת. הסרטון מראה בצורה ברורה ומקצת יותר את אונפ הפעלה, ניתן לראות את התזוזה של המנוע SERVO ואת ריצת הרדר במסק השני.



שני המסכים עובדים ב프וטוקול SPI لكن היו לי שני רגליים משותפות בשני המסכים. בעזרת שני הרגליים האלו המסכים מתקשרים בפרוטוקול SPI עם המיקרו בקר. ניתן לראות הסבר מוחכ בנווגע ל프וטוקול זה בספר שלי תחת הכוורת "פרוטוקול SPI"

קישור לסרטון: <https://youtube.com/shorts/rToKhwszY90?feature=share>

## ביבליוגרפיה:

במהלך הפרויקט שלי נזرتني במספר מקומות באינטרנט כדי לרכז ולסכם את המידע בקורס הכי פרקטית ומקצועית שאפשר. נזرتني במספר אתרים.

האתרים שיצא לי ללמידה ולרכז מידע מדויק שעזר ליקדם את הספר פרויקט היו אתרים שעוזרתי בהם בכל תקופה הלימודים שלי במהלך התואר הטכנולוגי (יג-יד) שעשיתי.

באתרים אלו ישנו סיכומים רבים ו מבחני תרגול שהייתי מתכוון דרכם ומתפתח באופן כללי כדי לחזק ולשפר את הידע שלי לגבי חישנים למיניהם או פרוטוקולים שונים.

לדעתי גם במהלך התואר ובאופן כללי בחיים יש צורך להתפתח מחו למסגרת הלימודים הקבועה לנו ולחקות קורסים עצמאיים כדי לקבל ידע ממקורות נוספים.

### האתרים העיקריים בהם הם:

- האתר לאלקטרוניקה ומחשבים אתר גדור עם סיכומים וחבים בכל התחומיים. דרך האתר זה למדתי עוד בלימודי במהלך התיכון והוא עזר לי המון כדי לחזור ואךקדם את הידע שלי בכל תחום עלילות האלקטרוניקה, התכונות והמחשבים, קישור לאתר: <https://www.elec4u.co.il>
- אתר נוסף שעזר לי לגבי הפרויקט שלי גם במהלך הפרויקט בסוף לימודי בתיכון במקצוע אלקטרוניקה ומחשבים וגם בלימודי בתואר הטכנולוגי במהלך זהו האתר של פורט, יש לו חומר לימודי, סיכומים, דוגמאות ועוד הרבה דברים שימושיים שיכולים לעזור להבנת הנושאים בכל עלילות האלקטרוניקה, התכונות ומחשבים. הקישור לאתר שלהם: <https://www.arikporat.com>
- ספר טוב מאוד שמננו לקחתי חלק מההטבות ויש בו Data-sheets I2C בנושא זה- ודברים טובים ו שימושיים להבנת הפרוטוקול תקשורת I2C. הקישור בספר זה שעוזרתי בו הוא: <https://www.st.com/resource/en/datasheet/vl53l8cx.pdf>
- מקום נוסף שהשתמשתי בו על מנת לסטם מידע ולקיים חלק מההטבות בנוגע למגוון SERVO זהו הספר עם הקישור הבא: <https://www.arikporat.com/wp-content/uploads/2024/03/%D7%9E%D7%A0%D7%95%D7%A2-SERVO.pdf>
- נזرتني בדף הבא על מנת לסטם לגבי החישן אוור שהוא חלק מן המודול חישן LDR שבו אני משתמש בפרויקט שלי, הקישור לאתר הוא: <https://gabbyshimoni.wixsite.com/arduino-programming/ldr>
- כדי להעלות את האתר שלי השימושי בשני תוכנות הראשונה GitHub שבה עשית העליתי את התקינה שבה נמצאים הקודם CSS+HTML בנוסח לכך שם נמצאים התמונות שקישרתי לאתר לסרטונים הראשיים בעמוד הבית. ובנוסף הקובץ PDF של הספר שלי, על מנת שנitin יהיה לפתוח אותו באתר או Netlify צריךゾהות את מיקום קובץ ה-PDF ולכון הכנסתי אותו לתיקייה הראשית שמנה אני מעלה את כל המסמכים שלו ל- GitHub וב- Netlify אני למעשה מקשר את התקינה שמאוכסנת ב- GitHub ולאחר מכן יוצר את הדומין של האתר (ה קישור) שדרכו למעשה אני יכול להכנס אל הדף האינטרנט שלי.

[קישור לאתר שלי](https://raybrook-project-santibot15.netlify.app):